Advances in Enterprise Control

Symposium Sponsored by JFACC Program, DARPA-ISO

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	15-16 Nov 99	Final

4. TITLE AND SUBTITLE
AEC Proceedings (Advances in Enterprise Control Symposium Sponsored by JFACC Program, DARPA-ISO)

5. FUNDING NUMBERS

6. AUTHORS
Multiple

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Multiple

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency - Information Systems Office
Col Dan McCorry, Program Manager for the JFACC Program
3701 N. Fairfax Dr
Arlington VA 22203-1714

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views, opinions, and/or findings contained in these papers does not constitute endorsement by DARPA nor the Department of Defense of the information contained therein

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited. Duplication or reproduction of these documents, in whole or in part, in hardcopy or electronic form, requires acknowledgement of DARPA-ISO, the authors, and their affiliated organizations

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document contains copies of 33 papers prepared for and presented at the November 1999 DARPA JFACC Symposium on Advances in Enterprise Control.

The purpose of the Symposium was to bring together researchers and practitioners from industry, government and academia to present and discuss the latest developments in all aspects of enterprise control. The Symposium presented papers that (a) describe the results of original research on the topics of interest, (b) provide broad reviews of the state-of-the-art, and (c) propose and advocate new research directions. Also presented were papers that describe significant practical experiences with current enterprise control systems: complex dynamic phenomena, non-obvious successes and failures, requirements and unmet needs.

The modern enterprise is a large-scale dynamic system with broadly distributed and potentially conflicting goals, resources and constraints, with multiple semi-autonomous participants of both human and artificial nature (e.g., large military operations, financial/trading institutions, logistics systems, manufacturing plants, power grids). The increasing capabilities of technology to collect, automatically generate, and disseminate information offer the possibility for large-scale enterprises to be more responsive to change. Enterprise plans and orders quickly become obsolete as new information about the current situation becomes available. The challenge is to use real-time information to re-direct enterprise operations effectively. Such systems and challenges define the scope of the Symposium.

14. SUBJECT TERMS
modeling and representations of large-scale enterprise control systems
reasoning at multiple levels of abstraction
controllability and stability
adaptive control in large-scale enterprises
game-theoretic control and adversarial control
role and impact of humans on the control loop
distributed and multi-agent control architectures
observation and state estimation
failures and pathological behavior in enterprise control
requirements of and for enterprise control systems

15. NUMBER OF PAGES
297

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Advances in Enterprise Control
Symposium Proceedings

Sponsored by:

JFACC Program,
Defense Advanced Research Programs Agency (DARPA)
Information Systems Office (ISO)

San Diego, CA
November 15-16, 1999

Program Committee

Preface

The 1999 DARPA-JFACC Symposium on Advances in Enterprise Control was organized under the sponsorship of the Joint Force Air Component Commander (JFACC) Program in the Information Systems Office (ISO) of the Defense Advanced Research Projects Agency (DARPA). The purpose of this symposium was to bring together researchers and practitioners from industry, government and academia to present and discuss the latest developments in all aspects of enterprise control. The participants of the Symposium presented papers that (a) described the results of original research on enterprise control, (b) provided broad reviews of the state-of-the-art theory and techniques, and (c) proposed and advocated new research directions.

The modern enterprise is a large-scale dynamic system with broadly distributed and potentially conflicting goals, resources and constraints, with multiple semi-autonomous participants of both human and artificial nature (e.g., large military operations, financial/trading institutions, logistics systems, manufacturing plants, power grids, etc.). The increasing capabilities of technology to collect, automatically generate, and disseminate information offer the possibility for large-scale enterprises to be more responsive to change. Enterprise plans and orders quickly become obsolete as new information about the current situation becomes available. The challenge is to use real-time information to re-direct enterprise operations effectively. Such systems and challenges defined the scope of the Symposium.

Table of Contents

Section 1

Theory and Techniques for Control Systems

Robust/Game-Theoretic Methods in Filtering and Estimation

William M. McEneaney *

Abstract

Robust/Game Theoretic approaches to filtering and estimation are considered. Estimates are obtained where the errors are bounded by some measure of the cost to the opponent. Both continuous and discrete models are considered. In the continuous case, a rather fast algorithm is discussed for nonlinear problems, and it is noted that a Riccati equation update is sufficient in the linear–quadratic sub-case. Analogous results are presented in the discrete case, although the additional technicalities are not present there.

1 Introduction

We present a Robust/Game Theoretic approach to the filtering problem. In this approach, one obtains an estimate for the state of some system and a certain bound on the error in this estimate in terms of some cost associated with the disturbances. In situations where the largest disturbances are effected by an opposing player in a game, this is a natural approach in that this cost can represent some actual cost to the opposing player. However, this approach can be used in a more general context. In fact, the natural control analogue is H_∞ control, which has the game interpretation (also dissipation interpretation) of bounding the measure of the poorness of the control by a measure of the size of the disturbances in the system (see for instance [1], [10] among many notable others). This approach has been used with significant success in recent years. Further, it has recently been shown that the nonlinear H_∞ problem is equivalent to the risk–averse limit of a risk–sensitive stochastic formulation of the problem (cf. [3]) thus yielding a stochastic interpretation.

This approach is especially appropriate for systems where one desires to track the state of some process in the presence of an antagonistic player. In such a case, the traditional approach of assuming random disturbances generated by, say for instance a Gaussian process, would not be appropriate. It is also appropriate when the estimates will be utilized by a significantly risk–averse controller.

We will develop the Robust/Game Theoretic filtering approach first in the case of a continuous time/continuous state process with measurements at discrete times. This will lead to a PDE (partial differential equation) formulation for the information state between measurement times, and a simple subtraction for the measurement update. The information state [7] is the analogue of the conditional density in the stochastic approach. Some relatively fast methods have been developed for this problem in the fully nonlinear case, however these are still only appropriate for low-dimensional (sub-)systems. We will also briefly indicate the Riccati equation formulation which may be used in the linear–quadratic case; this is similar to the linear–quadratic Kalman filter update.

We will also discuss the development in the case of a discrete time/ discrete state–space model. Here, the technical difficulties will be significantly reduced. Readers who are interested only in the discrete case should be able to proceed directly to Section 4.

2 Continuous Models

Portions of this section are condensed from material in [9], but are needed here for the material to follow. Let the state dynamics be given by

$$\frac{dX}{dt} = f(X) + \sigma(X)w \tag{1}$$

where X is the state (taking values in $\mathbf{R}^n$), f represents the nominal dynamics, w is a deterministic, but (a priori) unknown, L_2 process, and σ is some (matrix–valued) multiplier on the disturbance. Assume that at each measurement time t_i, we receive a measurement which we model as

$$z_i = g(X(t_i)) + \rho(X(t_i))v_i \tag{2}$$

*Dept. of Mathematics, North Carolina State University, Raleigh, NC 27695–8205, USA, http://www4.ncsu.edu/~wmm/, wmm@eos.ncsu.edu, Research partially supported by DARPA grant F30602-99-2-0548, AFOSR grant F49620-98-1-0304 and NSF grant DMS-9971546.

where g represents the disturbance–free measurement model (taking values in $\mathbf{R}^l$), and v_i is the disturbance at time t_i. Here z_i may be vector–valued, and ρ may be matrix–valued.

We will assume that f, σ, g and ρ are all continuously differentiable. We will assume that f, g and σ are globally Lipschitz in x, that σ is bounded, and that $a \doteq \sigma\sigma^T$ is uniformly non–degenerate, i.e. that there exists $\delta > 0$ such that

$$\xi^T\sigma(x)\sigma^T(x)\xi \geq \delta|\xi|^2 \qquad \forall\, x \in \mathbf{R}^n,\ \forall \xi \in \mathbf{R}^n. \quad (3)$$

We will also assume that $\text{Range}(\rho(x)) = \mathbf{R}^l$ for all $x \in \mathbf{R}^n$ which guarantees that for any z, x there exists some v satisfying (2) which should certainly be the case if our measurement model is properly constructed. Finally, we define ρ^{-1} by

$$\rho^{-1}(x)b = \text{argmin}\{|v| : \rho(x)v = b\}. \quad (4)$$

Assume that ρ^{-1} is uniformly bounded, that is, there exists $C_\rho < \infty$ such that $|\rho^{-1}(x)b| \leq C_\rho|b|$ for all $x \in \mathbf{R}^n$ and $b \in \mathbf{R}^l$. Note also that these assumptions imply that if we view the integral version of (1)

$$x_T \doteq X(T) = x_0 + \int_0^T f(X(t)) + \sigma(X(t))w(t)\,dt$$

as a mapping from x_0 to x_T then this mapping is one–to–one and onto for any $w \in \text{L}_2$, thereby ensuring that we may invert it.

Let $p_0(x_0)$ be a measure of our uncertainty about the initial state x_0, and let it have at most quadratic growth, so that $0 \leq p_0(x) \leq C(1 + |x|^2)$ for all $x \in \mathbf{R}^n$ for some $C < \infty$.

Suppose that we wish to estimate the state at time T, and that there have been N measurements in $(0, T]$. Consider the cost criterion

$$J(T, x_T, w(\cdot)) = -p_0(x_0) - \int_0^T |w(t)|^2\,dt - \sum_{i=1}^N |\rho^{-1}(X(t_i))[y_i - g(X(t_i))]|^2, \quad (5)$$

and corresponding information state

$$P(T, x_T) = \sup_{w \in L_2} J(T, x_T, w).$$

Note that P is a generalization of the (additive inverse of the) minimum disturbance energy required for the state to be x_T at time T while producing the measurements observed up to time T.

Due to the discrete nature of the measurements, it is helpful recall the form of the dynamic programming principle for such a system. Let $P(t_j^+, x)$ be the value at time t_j just after measurement j, and $P(t_j^-, x)$ be the value at time t_j just before the measurement. If T is not a measurement time, and $t \in (t_{k-1}, t_k)$, we have

$$P(T, x) = \sup_{w \in L_2([t,T];\mathbf{R}^m)} \left\{ P(t, X(t)) - \int_t^T |w(r)|^2\,dr - \sum_{j=k}^N |\rho^{-1}(X(t_j))[z_j - g(X(t_j))]|^2 \right\},$$

and when T occurs at the time of measurement j, we have

$$P(T^+, x) = P(T^-, x) - |\rho^{-1}(x)[z_j - g(x)]|^2.$$

The corresponding dynamic programming equations are

$$V(0, x) = -p_0(x) \qquad t = 0 \quad (6)$$

$$0 = V_t - \sup_{w \in \mathbf{R}^m} \{-[f(x) + \sigma(x)w]^T\nabla_x V - |w|^2\}$$

$$= V_t + f^T(x)\nabla_x V - \frac{1}{4}(\nabla_x V)^T a(x)\nabla_x V \quad (7)$$

$$V(t_j^+, x) = V(t_j^-, x) - |\rho^{-1}(x)[z_j - g(x)]|^2 \quad (8)$$

The following result is a standard type of dynamic programming result clarifying the (weak) sense in which the information state is a solution of PDE (7). However, as will be seen in the next section, there is a new, fast numerical technique which does not directly use the PDE interpretation as the basis for the algorithm. A proof of the theorem may be found in [9].

Theorem 2.1 *P is a continuous viscosity solution of (7) between measurement times.*

We will now indicate how the information state, P, may be used to obtain a robust/game theoretic state estimate. First we will indicate a direct approach which was originally proposed by Mortensen ([6], [11]), and then we will combine the above with a quadratic estimation criterion to obtain the game theoretic filter. For the direct Mortensen approach, note that $-P(T, x)$ represents, in some sense, the minimum disturbance energy needed for the target state at time T to be x for given measurements z.. The Mortensen estimate is any

$$\widehat{e}_T \in \underset{x \in \mathbf{R}^n}{\text{argmax}}\, P(T, x). \quad (9)$$

We can assert the existence of the argmax in (9) under reasonable assumptions as follows.

Theorem 2.2 *Suppose there exists $C_p > 0, \overline{x}_0 \in \mathbf{R}^n$ such that $p_0(x) \geq C_p|x - \overline{x}_0|^2$ for all $x \in \mathbf{R}^n$. Then $\widehat{e}_T$ exists.*

PROOF. Let the global Lipschitz constant on f be K, and let the bound on σ be m_σ. Let $\overline{x}_T \in \mathbf{R}^n$ be such that $\overline{X}(0) = \overline{x}_0$ when $\frac{d\overline{X}}{dt} = f(\overline{X})$ and $\overline{X}(T) = \overline{x}_T$ (the existence of which is guaranteed by the assumptions). Let X satisfy the dynamics of (1) with $X(T) = x_T$ for any $w \in L_2[0,T]$ and any $x_T \in \mathbf{R}^n$. Let $x_0 \doteq X(0)$. Then

$$|X(t) - \overline{X}(t)| = |(x_0 - \overline{x}_0) + \int_0^t [f(X) - f(\overline{X}) + \sigma(X)w]\, dr|$$

which by the assumptions and Cauchy-Schwarz

$$\leq |x_0 - \overline{x}_0| + m_\sigma\sqrt{T}\|w\|_{L_2} + \int_0^t K|X - \overline{X}|\, dr$$

where K is the Lipschitz bound on f, and m_σ the bound on $\sigma(x)$. Employing Gronwall's inequality yields

$$|x_T - \overline{x}_T| \leq (|x_0 - \overline{x}_0| + m_\sigma\sqrt{T}\|w\|_{L_2})C_T$$

for appropriate $C_T < \infty$. Consequently,

$$C_p|x_0 - \overline{x}_0|^2 + \|w\|_{L_2}^2 \geq C_p\left[\frac{|x_T - \overline{x}_T|}{C_T} - m_\sigma\sqrt{T}\|w\|_{L_2}\right]^2 + \|w\|_{L_2}^2$$

which after some calculations one finds

$$\geq \frac{C_p}{C_T^2(1 + m_\sigma^2 T)}|x_T - \overline{x}_T|^2.$$

Therefore, by (5) and the assumptions

$$J(T, x_T, w) \leq -\frac{C_p}{C_T^2(1 + m_\sigma^2 T)}|x_T - \overline{x}_T|^2.$$

Since this is true for any $w \in L_2$, one has

$$P(T, x_T) \leq -\frac{C_p}{C_T^2(1 + m_\sigma^2 T)}|x_T - \overline{x}_T|^2 \qquad (10)$$

which yields the result. □

Although the direct Mortensen estimate is of some interest, the following robust/game theoretic estimate will have more desirable properties in this context. Specifically, the information state can be combined with a quadratic estimate error criterion to obtain the robust filter. Let the dynamics and measurement models be those given above in (1) and (2), and make the same assumptions. However, now let the cost criterion take the form

$$I(T, e, x_T, w) = \gamma^2 J(T, x_T, w) + |x_T - e|^2.$$

Let

$$W(T, e) = \sup_{x_0 \in \mathbf{R}^n} \sup_{w \in \mathcal{W}} I(T, \epsilon, x_T, w) = \sup_{x_T \in \mathbf{R}^n} \sup_{w \in \mathcal{W}} I(T, e, x_T, w).$$

Note that

$$W(T, \epsilon) = \sup_{x_T \in \mathbf{R}^n} \left[\gamma^2 P(T, x_T) + |x_T - e|^2\right]. \qquad (11)$$

It is clear here that γ must be large enough so that this supremum will be finite and achieved at some x_T. By (10), one obtains the following.

Theorem 2.3 *There exists $\gamma < \infty$ such that the supremum in (11) is finite.*

This lower bound on such γ is directly analogous to the optimal disturbance attenuation parameter in H_∞ control below which the supremum in that problem becomes unbounded. Note that in a computational system, given a P, one can choose a γ corresponding to this P such that the supremum is finite.

Now note that since $W(T, e)$ is a supremum of functions which are strictly convex in e, it is also strictly convex in e. Further, $W(T, e) \to \infty$ as $|e| \to \infty$. Consequently, the minimum over ϵ is obtained at some point, and we define the robust filter estimate uniquely at time T as:

$$e_T \doteq \operatorname*{argmin}_{e \in \mathbf{R}^n} W(T, e).$$

Further, by (10) one may choose γ large enough so that $\min_{\epsilon \in \mathbf{R}^n} W(T, e) \leq 0$, in which case one obtains the error estimate

$$|x_T - e_T|^2 \leq \gamma^2 \left[p_0(x_0) + \|w\|^2 + \sum_{i=1}^N |v_i|^2\right]$$

for all $x_0 \in \mathbf{R}^n$ and all $w \in \mathrm{L}_2$ where $\|\cdot\|$ represents the L_2–norm over $[0, T]$. This is the robust bound on the estimate error in terms of the energy of the disturbance.

Note that the information state computations are recursive. That is, having obtained an estimate at time T, one can obtain an estimate at time $\widehat{T} > T$ by extending the solution of (9) from T to $\widehat{T}$.

3 Computational Issues for the Continuous Model

The chief difficulty in a general method for the computation of this nonlinear filter is the propagation of the information state, P, between measurement times. In the

stochastic analogue, this is similar to the propagation of a solution to the Zakai or Kushner equation. The form of the above problem leads to a unique method for the propagation of the information state between measurement times. This method allows for greatly increased computational speed. The following discussion outlines the analytical basis of the algorithm; the details may be found in [2].

Let S_t denote the semi-group associated with PDE (7). That is, given the solution at time t, $P(t,\cdot)$, the solution at time $t+\tau$ is given by $S_\tau[P(t,\cdot)](x)$. Consider the max–plus algebra over $\mathbf{R}\cup\{-\infty\}$ which is given by

$$a \oplus b = \max\{a,b\}$$
$$a \otimes b = a + b.$$

An important result is that this semi-group is linear over the max–plus algebra. That is, given functions $\phi_1(x), \phi_2(x)$ and constants a_1, a_2, one has

$$S_t[(a_1 \otimes \phi_1(\cdot)) \oplus (a_2 \otimes \phi_2(\cdot))](x) = \big(a_1 \otimes S_t[\phi_1(\cdot)](x)\big) \oplus \big(a_2 \otimes S_t[\phi_2(\cdot)](x)\big).$$

Using a max–plus basis expansion of the space of solutions, so that $P(t_i,x) = \bigoplus_j [a_j(t_i) \otimes \psi_j(x)]$ with basis functions ψ_j, one may obtain

$$S_\tau\Big[\bigoplus_j [a_j(t_i) \otimes \psi_j]\Big](x) = \bigoplus_j [a_j(t_i) \otimes S_\tau[\psi_j]](x).$$

Precomputing the $S_\tau[\psi_j]$, and expanding them in the form $S_\tau[\psi_j] = \bigoplus_k [B_{j,k} \otimes \psi_k]$, one obtains the following max–plus linear propagation algorithm

$$A(t_i + \tau) = B \otimes A(t_i)$$

where $A(t)$ is the vector of coefficients in the basis expansion at time t and B is the matrix $[B_{j,k}]$. Note that the matrix B can be precomputed, and consequently, the (approximate) propagation of P between measurement times reduces to a max–plus matrix–vector multiplication. See [2] for the details.

Some snapshots of P in a simple one–dimensional example are depicted in Figures 1–4 to indicate the possible nonsmooth and multi-lobe nature of the information state. (Each figure actually depicts three closely spaced snapshots.)

Even with the significant advance described just above, one cannot hope to filter high-dimensional nonlinear problems in real–time. Of course, in the linear–quadratic case, the computations reduce to propagation of the Riccati equations as usual, and so are quite computable. To see this, suppose that $f(x) = Ax$, $g(x) = Bx$, σ, ρ are constant, and $p_0(x) = -(x-\overline{x}_0)^T \widetilde{P}_0 (x-\overline{x}_0)$ for some $\overline{x}_0$ and positive definite symmetric $\widetilde{P}_0$. Then, one finds that $P(t,\cdot)$ is quadratic. Specifically, the information state may be written in the form $P(t,x) = -(x-\overline{x}(t))^T \widetilde{P}(t)(x-\overline{x}(t)) + r(t)$ for proper choice of

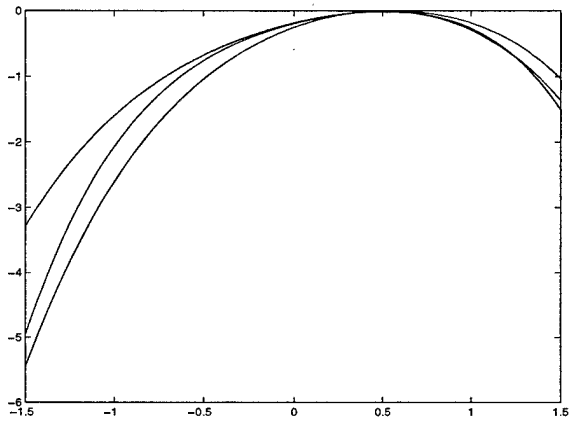


Figure 1: $(t = 0)$

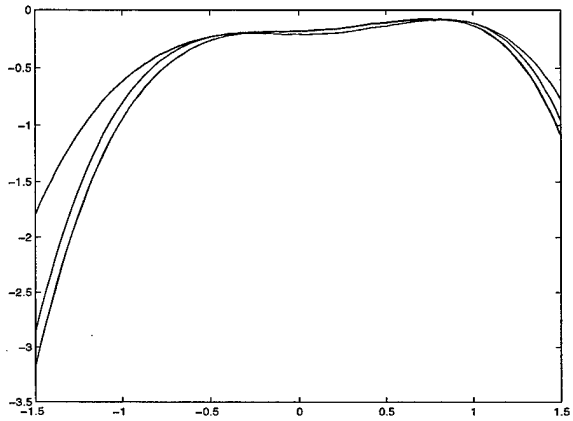


Figure 2: $(t = 0.2)$

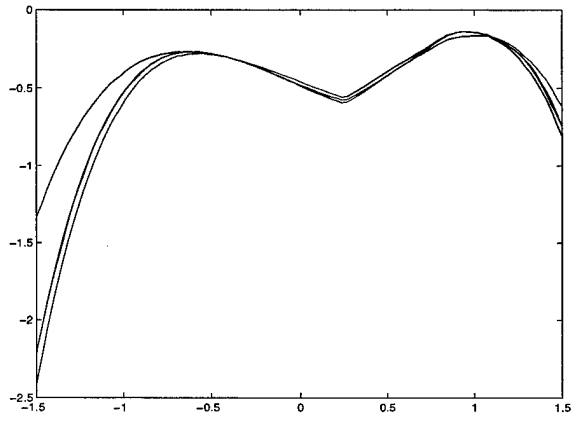


Figure 3: $(t = 0.4)$

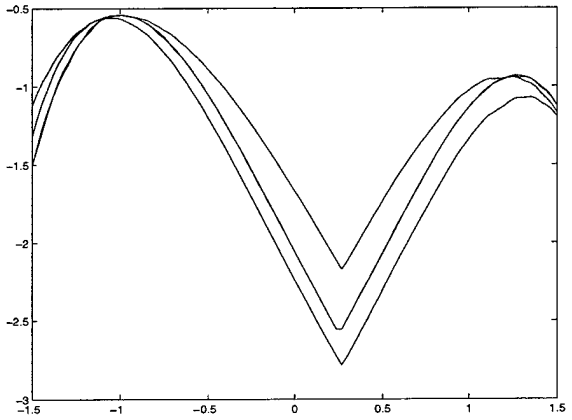


Figure 4: ($t = 0.6$)

$\widetilde{P}, \overline{x}, r$. In fact, between measurement times, $\widetilde{P}$ satisfies

$$\dot{\widetilde{P}} = -(A^T\widetilde{P} + \widetilde{P}A) + \widetilde{P}^T\sigma\sigma^T\widetilde{P},$$

while at measurement times, the update is given by

$$\widetilde{P}^+ = \widetilde{P}^- - B^T\rho^{-1}\rho^{-T}B.$$

There are corresponding updates for $\overline{x}(t)$ and $r(t)$. Further this estimate is identical to that generated by the (now linear) Kalman filter for the analogous (linear) stochastic model with initial covariance $\widetilde{P}_0^{-1}$, and in fact, $\widetilde{P}^{-1}(t)$ is the covariance matrix for $t > 0$.

4 Discrete Time/Discrete State

The previous sections described some of the current theory behind robust/game-theoretic estimation in the case of a system where the variables were continuous. Of course in many situations such as the "command and control" situation, many of the variables are expected to be discrete, say from some finite set. Here we will suppose the state variables take values in a discrete set, say $\mathcal{M}$. The typical approach to such problems is the Markov chain approach. In contradistinction to this, we will indicate a robust/game-theoretic approach to the estimation problem with state taking values in $\mathcal{M}$. It is not difficult to imagine applications for which this approach would be appropriate. First of all, there are situations where the dynamics and measurements may be corrupted by an antagonistic player. For instance, in battlefield management, one may have an opposing player who will attempt to corrupt our observations so as to give a false picture of the true situation. Secondly, there are problems where one wishes to be very risk-averse such as certain portfolio optimization problems.

To further clarify the problem formulation here, let us suppose that the time variable is discrete (although it will be clear that continuous time problems have an analogous approach). In fact, let us simply suppose that the time variable takes integer values starting from initial time $k = 0$. In this section, we will ignore the technical details in order to focus simply on the general picture.

Let the state at time k be given by α_k where $\alpha_k \in \mathcal{M}$ for all k. Let up suppose that the cardinality of $\mathcal{M}$ is m, and let the possible values of the state correspond to the elements of $\{1, 2, 3, ..., m\}$ which we will henceforth refer to as $\mathcal{M}$ (i.e. we will simply take $\mathcal{M} = \{1, 2, 3, ..., m\}$).

By analogy with the continuous case, one could suppose a dynamic model of the form

$$\alpha_{k+1} = f(k, \alpha_k, w_k)$$

where w_k is the disturbance (due to an antagonistic player) at time k. Then, one could associate a cost with each value of w_k of the form say $D(w_k)$. However, it appears conceptually and numerically simpler to develop a model more analogous to the Markov chain approach. For instance, by analogy with the Markov chain transition **probability** matrix, we will formulate the above dynamics and cost in terms of a transition **cost** matrix. more specifically, suppose the cost for transitioning (in one time-step) from state $i \in \mathcal{M}$ to $j \in \mathcal{M}$ is denoted by $c_f(i, j)$ where we are further simplifying the notation by assuming that this is time–independent. Then, we form the matrix C^f as the matrix whose elements are given by

$$C^f_{i,j} = c_f(i, j).$$

Note that one may restrict the state from transitioning from say state i_0 to state j_0 in one time step by setting $C^f_{i_0,j_0} = +\infty$.

Similarly, suppose that we have a fixed, finite set of possible measurement values. Let us suppose that there are n possible values, and simply denote them by $\mathcal{N} = \{1, 2, 3, ..., n\}$. Then for each time k, the k^{th} measurement takes its value in $\mathcal{N}$, i.e. $y(k) \in \mathcal{N}$ for all k. Let the cost to the disturbance for producing measurement j when the state is i be $c_g(i, j)$. Again, we can organize this data in the form of an $m \times n$ matrix C^g where

$$C^g_{i,j} = c_g(i, j).$$

Finally, let the vector p^0 of length m correspond to our initial information (at time $k = 0$).

It is not difficult to see how one would compute the worst–case cost for being in state j at time $k = 1$ given measurement $y(1) \in \mathcal{N}$. It would be

$$\sup_{i \in \mathcal{M}} \left\{ -\left[p^0_i + C^f_{i,j} + C^g_{j,y(1)} \right] \right\}.$$

To generalize this to later times, let $\widehat{\alpha}^k$ be a vector of length $k+1$ of elements of $\mathcal{M}$ corresponding to a state transition path from time 0 to time k. Define the cost to be in state j at time k with past history $\widehat{\alpha}^{k-1}$, given measurements $y(l)$ for $l = 1, 2, ...k$ to be

$$J(k, j, \widehat{\alpha}^{k-1}) = -\Bigg[p^0_{\widehat{\alpha}^{k-1}_0} + \sum_{l=1}^{k-1} \Big(C^f_{\widehat{\alpha}^{k-1}_{l-1}, \widehat{\alpha}^{k-1}_l} + C^g_{\widehat{\alpha}^{k-1}_l, y(l)} \Big) + C^f_{\widehat{\alpha}^{k-1}_{k-1}, j} + C^g_{j, y(k)} \Bigg].$$

As in the continuous case, one defines the "information state" at time k for state j in the worst–case cost sense as

$$P(k, j) = \sup_{\widehat{\alpha}^{k-1} \in \mathcal{A}^{k-1}} J(k, j, \widehat{\alpha}^{k-1})$$

where $\mathcal{A}^k$ is the set of sequences of length $k+1$ taking values in $\mathcal{M}$.

The information state captures all the information needed to produce the robust estimate at the current time. (It is analogous to the conditional probability distribution in stochastic models.) In some sense, the information state, $P(k, j)$, represents some generalized notion of the cost to the opposing player for the state to be in state j at time k while having produced the measurements that have been seen. Given the information state at any time, one chooses the parameter γ sufficiently large so that

$$W(k, e) \doteq \sup_{j \in \mathcal{M}} \left[|j - e|^2 + \gamma^2 P(k, j) \right]$$

has a minimum (in e) which is non-positive. (Note that the existence of such a γ actually implies assumptions on the choice of C^f, C^g, p^0. In the continuous case of Section 2, there were standard simple assumptions which guarantee this. However, in the more "free–form" description of this section, there are no obvious choices for these assumptions, and one would need to be more specific about the form of the model before making such assumptions. Also, one might choose something other than the quadratic form $|j-e|^2$ since there is no obvious reason why that would necessarily be the most appropriate form.) Let e_k be the minimizer. This implies that

$$W(k, e_k) = \sup_{j \in \mathcal{M}} \left[|j - e_k|^2 + \gamma^2 P(k, j) \right] \leq 0.$$

In other words,

$$|j - e_k|^2 \leq -\gamma^2 P(k, j) \qquad \forall\, j \in \mathcal{M}.$$

Using the definition of P, this yields

$$|j - e_k|^2 \leq \gamma^2 \Bigg[p^0_{\widehat{\alpha}^{k-1}_0} + \sum_{l=1}^{k-1} \Big(C^f_{\widehat{\alpha}^{k-1}_{l-1}, \widehat{\alpha}^{k-1}_l} + C^g_{\widehat{\alpha}^{k-1}_l, y(l)} \Big) + C^f_{\widehat{\alpha}^{k-1}_{k-1}, j} + C^g_{j, y(k)} \Bigg]$$

for any initial state $\widehat{\alpha}^{k-1}_0$ and any choice of hostile counteractions. That is the squared error in our estimate (e_k) of the state at time k is bounded above by the right-hand side where the right-hand side represents some measure of the costs to the opposing player for the disturbances it employs. Note that the larger one needs to take γ, the less one can attenuate the effects of the counteractions on our squared estimate error.

Let us also write down the dynamic programming equation that one would solve to update P from one time to the next.

$$P(k+1, j) = \sup_{i \in \mathcal{M}} \Big\{ C^f_{i,j} + C^g_{j, y(k+1)} + P(k, i) \Big\}.$$

Of course, the right-hand side is a max–plus linear operator on $P(k, \cdot)$, and this may be written more abstractly as

$$P(k+1, j) = \mathcal{S}_1 \big[P(k, \cdot) \big](j) \qquad \forall\, j \in \mathcal{M}$$

where $\mathcal{S}_1$ is max–plus linear.

We illustrate this with a very simple example. Let the state space be $\mathcal{M} = \{0, 1, 2, 3, 4, 5\}$. Let the transition cost matrix be

$$C^f = \begin{bmatrix} 0 & 1 & +\infty & +\infty & +\infty & +\infty \\ 0 & 0 & 1 & +\infty & +\infty & +\infty \\ 0 & 0 & 0 & 1 & +\infty & +\infty \\ 0 & 0 & 0 & 0 & 1 & +\infty \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Let the measurement cost matrix be

$$C^g = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 1 & 2 & 3 & 4 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 & 1 & 2 \\ 4 & 3 & 2 & 1 & 0 & 1 \\ 5 & 4 & 3 & 2 & 1 & 0 \end{bmatrix}$$

Let the initial information be $p^0 = (1, 0, 1, 2, 3, 4)$. A sequence of measurements was generated as depicted (by the $*$'s) in Figure 5 below. A robust estimator also appears (solid lines) in Figure 5. (This was just one possible robust estimator since we did not restrict ourselves to the quadratic form on the right-hand side in the definition of W.) The information state is depicted over the time period in Figure 6.

These estimates would then be used by a robust/game–theoretic controller. See [5],[4] for examples of the manner in which robust filters are sometimes used as controller inputs in the continuous case. We will be employing these discrete robust filter estimates in a simpler game–theoretic method to be discussed in full.

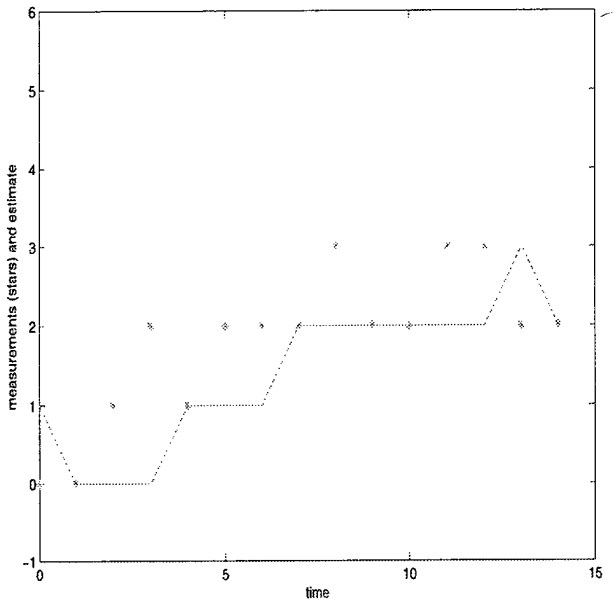


Figure 5: Measurements and Robust/Game–Theoretic Estimates

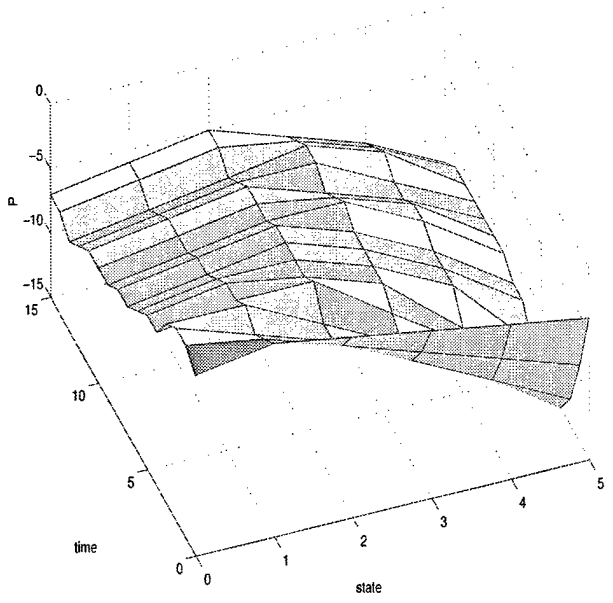


Figure 6: Information State ($P(k,j)$)

References

[1] T. Basar and P. Bernhard, **H_∞ –Optimal Control and Related Minimax Design Problems**, Birkhäuser, Boston (1991).

[2] W.H. Fleming and W.M. McEneaney, "A max–plus based algorithm for an HJB equation of nonlinear filtering", SIAM J. Control and Optim., (to appear).

[3] W. H. Fleming and W. M. McEneaney, Risk sensitive control on an infinite time horizon, SIAM J. Control and Optim., **33** (1995), 1881–1915.

[4] E. Gallestey, M.R. James and W.M. McEneaney, "Max–plus approximation methods in partially observed H_∞ control", 38th IEEE Conf. on Decision and Control (to appear).

[5] J.W. Helton and M.R. James, **Extending H_∞ Control to Nonlinear Systems: Control of Nonlinear Systems to Achieve Performance Objectives**, to appear, SIAM 1999.

[6] O. J. Hijab, Minimum Energy Estimation, University of California, Berkeley, PhD Thesis, 1980.

[7] M. R. James and J. S. Baras, Partially observed differential games, infinite dimensional HJI equations, and nonlinear H_∞ control, SIAM J. Control and Optim., **34** (1996), 1342–1364.

[8] V.P. Maslov, "On a new principle of superposition for optimization problems", Russian Math. Surveys, **42** (1987) 43–54.

[9] W. M. McEneaney, "Robust/H_∞ filtering for nonlinear systems", Systems and Control Letters, Vol. 33 (1998), 315–325.

[10] W. M. McEneaney, Robust control and differential games on a finite time horizon, Math. of Controls Signals and Systems, **8** (1995), 138–166.

[11] R. E. Mortensen, Maximum Likelihood Recursive Nonlinear Filtering, J. Optim. Theory Appl., **2** (1968) 386–394.

Solving Adversarial Control Problems with Abstract Board Games and Linguistic Geometry (LG) Strategies[1]

Boris Stilman

Computer Science & Engineering Dept.
University of Colorado at Denver
Campus Box 109, Denver, CO 80217-3364
e-mail: bstilman@cse.cudenver.edu
tel: (303) 556-3614
fax: (303) 556-8369;

Vladimir Yakhnis

Rockwell Science Center, M/S A30
1049 Camino Dos Rios
Thousand Oaks, CA 91360
e-mail: vryakhni@rsc.rockwell.com
tel: (805) 373-4856
fax: (805) 373-4862;

SUMMARY Linguistic Geometry (LG) is an approach to finding "good enough" strategies providing solutions for various kinds of abstract board games. We discuss the basic elements of LG and a small example illustrating how the LG strategies may be applied at a battlefield. Two sides, the Blues and the Reds are considered, each having an UAV bomber, a fighter and a ground installation to be protected. The LG strategies will aid the Blue commander in controlling the Blue UAV bomber and in selection a most beneficial behavior for the Blue fighter. They will also provide counteraction by showing the most beneficial moves for the Reds.

1. INTRODUCTION

Imagine a Global Operations Command and Control Center (GOCC). It plans, directs, and executes global planet-encompassing missions employing battlefield, naval, marine, air force, near-planet, and special operations (including Internet cyberwar). All the planning and control is based on the hierarchy of combat simulation models of various resolutions. Each model employs the LG strategies.

Linguistic Geometry (LG) is an approach to construction of mathematical models for knowledge representation and reasoning about large-scale multi-agent systems [9]. A number of such systems (called LG Complex Systems), including air/space combat, robotic manufacturing, software re-engineering, Internet cyberwar, etc., can be modeled as abstract board games. Those are multi-player games, whose moves can be represented by means of moving abstract pieces over locations on an abstract board. The dimensions of the board (2D, *n*D, and even non-linear space), its shape and size, the mobility of pieces, the player turns of making moves (including concurrent moves) – all can be tailored to model a variety of multi-agent systems. The purpose of LG is to provide strategies to guide the participants of a game to reach their goals. Traditionally, finding such strategies required searches in giant game trees. Such searches are often beyond the capabilities of modern and even conceivable future computers.

LG dramatically reduces the size of the search trees, thus making the problems computationally tractable. To achieve that, LG provides a formalization and abstraction of search heuristics of advanced experts in the form of the game strategies. Essentially, these heuristics replace the search by construction of such strategies. The formalized expert strategies yield efficient algorithms for problem settings whose dimensions may be significantly greater than the ones for which the experts developed their strategies. Moreover, these formal strategies proved to be able to solve problems from different problem domains far beyond the areas envisioned by the experts. Those strategies are not intended to provide solutions that are always optimal, but they are intended to provide "good enough" solutions. Although for some classes of problems, these formalized expert strategies yield provably optimal solutions, for the rest of the problems the LG strategies are nearly optimal. To formalize the heuristics, LG employs the theory of formal languages (i.e., formal linguistics), as well as certain geometric structures over the abstract board. Since both, the linguistics and the geometry, were involved, this approach was named Linguistic Geometry.

Let us return to the hierarchy of combat simulation models executed at GOCC. At the top (strategic) level, the lowest resolution model controls the global planet-

[1] This research is supported in part by a DARPA contract N66001-99-C-8509.

size operations, as well as the largest possible teams of military mobile units. The full spectrum of mobility of those teams is employed. In the LG terms, the LG "operational district" (an abstract board) is a low-resolution grid that embraces oceans, land, air, near-planet space, as well as Internet. The agents are friendly and opposing teams of submarines, ships, mobile military units on land, air force units, and space assault vehicles. The LG "reachability relations" reflect the mobility of all teams ranging from under-water sailing of submarines to orbit-changing maneuvers of assault satellites, to server-to-server leaps of anti-viruses. A hierarchy of higher resolution models control smaller teams and separate vehicles, while focusing on smaller operational districts like marine-land, air-land, or classes of space orbits.

A preliminary planning of operation with LG models is done by running multiple experiments in order to select the best Start State, i.e., the best initial configuration of all the friendly agents involved in the operation. After the actual engagement, the control is conducted in real time by multiple re-planning by taking into account actual advancement of agents, actual losses/gains, and changes of mobility. Similar planning and real-time control of operations is conducted on a smaller scale by each team and each military unit.

A variety of computers at the battlefield and at the GOCC run the LG Engine (LGE), a generic software platform tuned to combat simulations. All the computers are linked; they exchange data over the network. For the models of extremely high dimension, the LGE and the computer network may be reconfigured for the distributed parallel computations. The LGE requires different tuning for different levels of command and control. In the past, such tuning has been done by the military experts without the assistance of LG. In our imaginary scenario, the LGE would provide guidance for the experts throughout this tuning.

A global mission planned and executed by GOCC would require extensive coordinated actions including underwater maneuvering, space satellite hunting, and Internet cyberwar actions. A number of LG strategies and strategic patterns guiding this mission would have been retrieved from the LG Strategic Data Base (LG SDB). The main advantage of LG is the immediate generation of a strategy in the original situation. In addition, the LG strategies and patterns retrieved from the LG SDB would allow us to obtain more comprehensive solutions by significantly expanding the horizon during a limited time frame. Moreover, the retrieved strategies and patterns would allow us to verify strategies leading to familiar patterns of successful operations and to avoid strategies leading to known failures.

The LG SDB strategies have actually been developed not only during past military operations. The LG SDB is a growing collection of numerous brilliant strategies discovered in various problem domains including efficient re-construction of the legacy software and intelligent manufacturing. Many of those strategies are optimal. One of the theoretical results on optimality of LG strategies is included in this paper.

The new original strategies generated in the course of the global mission have been stored in the LG SDB. These strategies usually carry the significant knowledge and skills of the domain experts, ranging from Lieutenants to Chiefs of Staff. The knowledge of such experts has been stripped of unimportant details, formalized and included in the general LG framework at the stage of the LGE's tuning. The enriched LG strategies would also allow us to transfer this knowledge to a variety of different problem domains.

2. ABSTRACT BOARD GAMES (ABG) AND STRATEGIES

To solve the problem of systems control, we view the state transition process of a system as a contest/cooperation between several competing/ cooperating agents (or players), one of which may be Controlling-Agent (The Friend). When there are only two competing agents, we'll call the other agent (player) the Opposing-Agent (the Adversary). For simplicity sake, and also since the case of two competing agents is a common occurrence, we'll mostly limit our introductory discussion to that case.

Similar to the Controlling-Agent, the Opposing-Agent exercises at least a partial influence over the state transitions and, in contrast to the Controlling-Agent, its purpose is either to guide the controlled process to a point where the constraint would be violated or to prevent reaching the goal. Note that in these games concurrent moves by the players are permitted, in contrast to conventional games.

Within the ABGs area, the state is modeled by placement of various objects (called pieces or mobile agents) on an abstract board. The state transitions are defined by moving the pieces on the board, adding the pieces, or eliminating the pieces from the board. ABGs were defined in [11,9] by expanding on graph-games defined in [10,12].

A state strategy for a player A_i is an input-output automaton accepting the packet-moves of as input, and outputting moves for A_i. The input is used for memorizing some information about the play (limited by the memory of the automaton), whereas the output is used to guide the behavior of A_i during the play. A state-strategy for alternating two player games is illustrated on Figure 2.

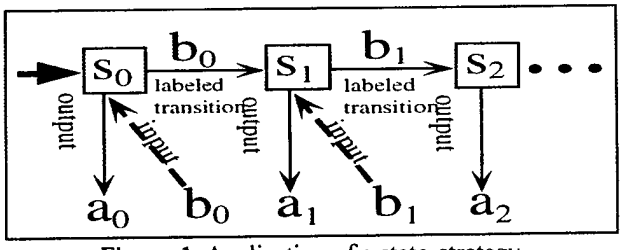


Figure 1. Application of a state-strategy

3. BASICS OF LG

The LG approach is applicable to a class of concurrent multi-agent systems. Consider two types of agents. Agents of the top level, the super-agents (also known as the players), are fully capable of acting by means a number of mobile entities called the (local) agents (or pieces). The environment may have a profound impact on the movements of agents. Some of the locations may be reachable in a certain number of steps, others may not be reachable at all. Consider systems with two super-agents that oppose each other. They are called the opposing sides. Usually, they pursue opposing goals. Each of them controls a team of local agents whose freedom of operation is restricted within the framework dictated by the respective player. The level of such restriction varies. On the one hand, the local agents may not have any freedom at all, i.e., each is fully controlled by the respective player. On the other hand, the LG tools are also applicable to models where the local agents are less constrained, and operate autonomously with some distributed intelligence.

In the following sections we shall describe a formal representation of a multi-agent system as an abstract board game, i.e., as a sub-class of Complex Systems.

3.1 Abstract Board Games as a Subclass of Complex Systems

In this section, we define a "discrete universe" by observing "the laws of discrete physics". The problems in such universe are very close to the board games like chess, checkers, etc. An abstract board or an area of the discrete universe, is represented by an arbitrary finite set X. Abstract pieces (also called the elements), represent the local agents standing or moving with a constant speed. We introduce such actions as movements, destruction of agents (removal from the system), collision, and collision avoidance. For the alternating concurrent (AC) and totally concurrent (TC) systems, we introduce concurrent movement.

DEFINITION 1. Complex System is the following eight-tuple:

$$\langle X, P, R_p, \text{SPACE}, val, S_i, S_t, TR \rangle,$$

where

$X = \{x_i\}$ is a finite set of *points* or locations of elements;

$P = \{p_i\}$ is a finite set of *elements*, (also called pieces) P is a union of two disjoint subsets P_1 and P_2 called the *opposing sides*;

$R_p(x, y)$ is a set of binary relations of *reachability* in X, where $(x, y) \in X$, $p \in P$);

val is a function on P with positive integer values describing the *values* of elements;

SPACE is the state space. A state $S \in$ SPACE consists of a partial function of *placement* of the form ON: $P \rightarrow X$ and of some additional parameters.

The value $ON_S(p) = x$ means that element p occupies location x at state S. When there is no confusion we will write ON instead of ON_S. To describe the function ON for a state S, we may write equations of the form $ON(p) = x$ (called the placement equations) for all elements p where ON is defined.

The additional parameters of a state may be various finite automata. E.g., the state of an *alternating* Complex System may include a two-valued automaton describing the player whose turn is to move at the state.

S_i and S_t are the sets of *initial* and *target* states, respectively. Often in case of only two players, S_t is the union of three sets, S_t^1, the winning states for the first player, S_t^2, the winning states for the second player, and S_t^1, representing the draw states.

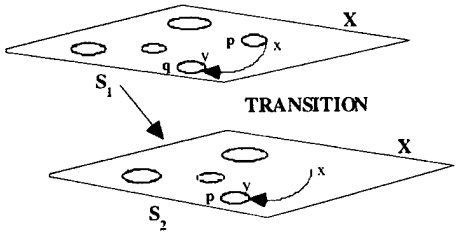


Figure 2. Complex Systems: TRANSITION(p,x,y)

TR is the set of all permissible state transitions. There are various means to describe the state transitions. Usually it is done via a collection of the game rules. E.g., a rule can be described by (1) the guard describing the applicability of the rule to the source state; (2) the *remove list* consisting of the placement equations of the pieces to be removed from the board; and (3) the *add list* consisting of the placement equations of the pieces to be added to the board. Finally, the guard may be described via an *applicability list* consisting of items pertaining to the individual pieces, e.g., of the form $ON(p) = x \wedge R_p(x,y)$.

In case of two players, the *goal* of each side could be to reach a state within its corresponding subset of target states, (i.e., S_t^1 or S_t^2) or, at least, a draw state from S_t^3. Thus the problem of finding the optimal operation of the System may be represented as a problem of searching for a sequence of transitions that would lead from the Start State in S_i to a target state in S_t and such that each side would make only the *best* moves.

Loosely speaking, a game strategy is an algorithm to select moves. *Solving a Complex System* means finding such a *strategy* for one side, that would guarantee that the desirable subset of target states, S_t^1, S_t^2, or S_t^3, will be reached (assuming that the other side makes arbitrary moves).

DEFINITION 2. Complex Systems may be divided into three classes:

An *Alternating Serial* (AS) system is the alternating Complex System where only one element at a time can move (plus some of the opposing elements located at the destination of this move are destroyed). The opposing sides alternate turns;

An *Alternating Concurrent* (AC) system is the alternating Complex System where all, some, or none of the elements of one side can move simultaneously (plus some the opposing elements located at the destinations of the concurrent moves may be destroyed). The opposing sides alternate turns.

A *Totally Concurrent* (TC) system is the Complex System where all, some, or none of the elements of both sides can move simultaneously or be destroyed.

The general class of Complex Systems (DEFINITION 1) does not include the concept of a block, i.e., several elements of the same side can occupy the same location at the same state. Also, an element can move to any reachable location. In that sense, an element can "cross over" any reachable location, whether it is occupied or not. Classes of Complex Systems where some of the elements are prohibited from crossing the points occupied by another element are defined in [9] as Complex Systems with blocked destinations and beams. General Complex Systems do not cover agents with variable speed. This class of agents is important for solving real world problems. A class of Complex Systems with elements with variable speed is introduced in [9]. Along these lines, the Complex Systems from DEFINITION 1 are the Complex Systems with *constant speed* which means that acceleration a = 0, for all elements and all states.

To apply the LG algorithm to various problem domains, we have to reflect all of the components of the Complex System: the operational district X, the set of mobile units P, their relations of reachability $R_p(x, y)$, etc. One of the most natural applications of LG is the class of *combat command and control* (combat c^2) problems where explicit mobile entities are organized in adversarial teams. LG allows to model land, undersea, air/space operations including near Earth and far space combat. Examples of Complex Systems for different problem domains are considered in [9].

To solve Complex Systems, we could use formal methods like those in theproblem-solving system STRIPS [1], nonlinear planner NOAH [4], or in other planning systems. In all cases, in order to find a strategy, we would have to search the state space employing, for example, mini-max search algorithms with alpha-beta pruning [3]. However, the size of the search space, especially, for real-world problems, though reduced by pruning, is still astronomical [2]. Thus, in practice, hardly any solution would be obtained for such complex problem as combat c^2.

3.2 A Survey of LG Tools

The formalization of abstract board games discussed in Section 3.1 is not unique. This class of problems could be formally represented via numerous existing techniques, where the majority of representations are equivalent. However, within LG, the chosen representation of the problem statement is coupled with the formal representation of the method used to solve it. The distinctive characteristic of the LG approach is that it serves as a foundation and a mathematical environment for the formal representation of the dynamic hierarchy of subsystems.

Within LG, we assume that each super-agent develops a model of the opposing side and it operates by the assumption that the adversary will do its best within this model. The model is used for planning the agent's actions and choosing the optimal one. The model also establishes local goals for local agents. These local goals are coordinated with the global goal of the corresponding super-agent. The system of local agents is decomposed into subsystems striving to attain these goals. For example, each second level subsystem includes local agents of both opposing sides: the goal of one side is to attack and destroy another side's local agent (a target), while the opposing side tries to protect it. In the robot control, for example, it means the following. (1) Selection of a pair of robots of opposing sides: one – as an attacking element, and the other – as a local target. (2) Generation of the local paths for approaching the target. (3) Generation of the paths of other robots supporting the attack or protecting the target. Achieving the local goals is intended to help the respective super-agent to achieve the global one. The model is dynamic: after every action, which may include concurrent movements of agents of

both sides, it is updated taking into account the new situation.

3.2.1 Trajectories.

The lowest level of the hierarchy of subsystems is a trajectory, i.e., a path, with an entity moving along it. When the entity moves forward, part of a trajectory that is left behind may disappear, and re-appear again, when an entity would backtrack during the search to explore another path. In LG trajectories are represented as strings of symbols.

A *trajectory* for an element p of P with the beginning at $x \in X$ and the end at $y \in X$ (where $x \neq y$) with the *length* l is the following formal string $t = \boldsymbol{a}(x_0)\boldsymbol{a}(x_1) \ldots \boldsymbol{a}(x_l)$ (with $x = x_0$ and $y = x_l$) over the alphabet $\{\boldsymbol{a}\} \times X$. The alphabet is further restricted as follows. Each successive point x_{i+1} is reachable from the previous point x_i, i.e., $R_p(x_i, x_{i+1})$ holds for $i = 0, 1, \ldots, l-1$; element p stands at the point x: ON(p) = x at a state S of the Complex System. A set of trajectories $\mathbf{L_t^H}(S)$ of the length less than or equal **H** is called the Language of Trajectories in state S within horizon **H**.

Formal tools, i.e., grammars of trajectories [9], generate various kinds of trajectories encountered in a number of problem domains, e.g., a shortest path, a path going around obstacles, etc.

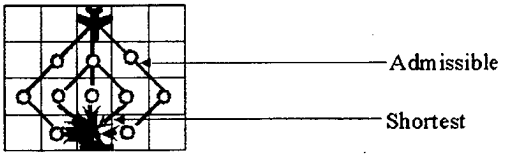


Figure 3. Shortest and admissible trajectories

3.2.2 Networks of Trajectories

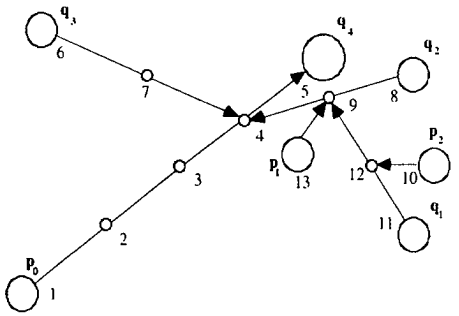


Figure 4. A network of trajectories

A trajectory of n^{th} degree consists of n shortest trajectories. Admissible trajectories are the trajectories of 2^{nd} degree. LG provides efficient algorithms for generating of trajectories. Usually, only the shortest and admissible trajectories are employed. Figure 3 illustrates some of the shortest and admissible trajectories employed in the example from Section 4.

At the higher levels of the hierarchy are networks of trajectories (with their entities). Every network has its ultimate goal related to a specific location, the final destination. A number of types of networks were introduced: webs, Zones which include attack Zones, domination Zones, retreat Zones, etc. Every network includes elements participating in achieving the ultimate goal. This participation is threefold. Elements of one side pursue their goal by moving, protecting each other, and destroying the enemies. Elements of the other side, by similar actions, do their best to prevent the opponent from achieving the goal. An example of such network is shown on Figure 4.

The basic idea behind these networks is as follows. Element p_0 should move along the main trajectory $\boldsymbol{a}(1)\boldsymbol{a}(2)\boldsymbol{a}(3)\boldsymbol{a}(4)\boldsymbol{a}(5)$ to reach the ending point 5 and remove the target q_4 (an opposing element). Naturally, the opposing elements should try to obstruct these movements by "dominating" the intermediate points of the main trajectory. They should come closer to these points (to point 4 in Fig. 3.1) and remove element p_0 after its arrival (at point 4). For this purpose, elements q_3 or q_2 should move along the trajectories $\boldsymbol{a}(6)\boldsymbol{a}(7)\boldsymbol{a}(4)$ and $\boldsymbol{a}(8)\boldsymbol{a}(9)\boldsymbol{a}(4)$, respectively, and wait (if necessary) on the next to last point (7 or 9) for the arrival of the element p_0 at point 4. Similarly, element p_1 of the same side as p_0 might try to obstruct the movement of q_2 by dominating point 9 along the trajectory $\boldsymbol{a}(13)\boldsymbol{a}(9)$. It makes sense for the opposing side to employ the trajectory $\boldsymbol{a}(11)\boldsymbol{a}(12)\boldsymbol{a}(9)$ of element q_1 to prevent this domination.

An element can participate in many networks simultaneously. With respect to this, the timing-related considerations with respect to the movements of elements are of significant import. For example, if an element has enough time to move along a trajectory to intercept an opposing element, this trajectory would be included in the network and the movement would be allowed. If this is not the case, the trajectory would be dropped. In a network, parts of trajectories, as well as bundles of trajectories, together with their entities may be eliminated when movement along them would not make sense with respect to the ultimate goal. This may be related to the timing, the state configuration, or other reasons. Calculations of time and other factors related to the network construction and re-construction are done within each network independently. For this purpose, the other networks would be ignored. A networks can be connected with another one if the ultimate goal of one of them becomes the goal of the joint network, while the goals of the components would become subordinate to the main goal. One of the major types of networks, the

attack Zone, is illustrated in Figure 4.

Networks may be formalized as strings of symbols over the alphabet $\{t\}\times P\times \mathbf{L}_t^{\mathbf{H}}(S)\times TIME\_BOUND$, where t is a symbol, $\mathbf{L}_t^{\mathbf{H}}(S)$ is the Language of Trajectories in state S within horizon **H**, P are elements of S, and TIME_BOUND is a finite set of time bounds τ measured by non-negative integers. Various types of networks are distinguished by the respective restrictions of this alphabet. Grammars of networks are considered in [9].

The attack Zone shown in Figure 4 is represented as follows:

$$Z = t(p_0,a(1)a(2)a(3)a(4)a(5),5)\ t(q_3,a(6)a(7)a(4),4)$$
$$t(q_2,a(8)a(9)a(4),4)\ t(p_1,a(13)a(9),3)$$
$$t(q_1,a(11)a(12)a(9),3)\ t(p_2,a(10)a(12),2)$$

Consider the movements of elements shown in Figure 4 in order to explain the reasoning behind the construction of this Zone. Assume that the goal of White is to remove target q_4, while the goal of Black is to protect it. Also, assume that Black and White alternate turns. Black starts first to move elements q_2 or q_3 to intercept element p_0. White follows in its turn by moving element p_0 to the target. Only those Black's trajectories are to be included in the Zone where the movement of the element makes sense, i.e., the *length of the trajectory is less than (or equal) the amount of time (time bound* τ*) allotted to it.* For example, the movement along the trajectories $a(6)a(7)a(4)$ and $a(8)a(9)a(4)$ makes sense, because they are of length 2 and time allotted equals 4: each of the elements has 4 time intervals to reach point 4 to intercept element p_0, assuming that Black starts first and p_0 would go along the main trajectory without delay. By definition, the trajectories of White's elements (except p_0) could only be of the length 1, e.g., $a(13)a(9)$ or $a(10)a(12)$. Because element p_1 can intercept element q_2 at point 9, Black's trajectory $a(11)a(12)a(9)$ of the element q_1 should be included: q_1 has enough time for movement to prevent this interception. The total amount of time allotted to the whole bundle of Black's trajectories connected (directly or indirectly) with a given point of the main trajectory is determined by the ordinal number of that point. For example, for point 4, it is equal to 4 time intervals. This number gives us the value of time bound t for the intercepting trajectories connected with the main trajectory at point 4.

An attack Zone consists of several kinds of trajectories:

- *the main trajectory*, representing a path for an agent of one of the players (called the main player of the Zone) to attack a vital target. E.g., $\mathbf{t}(p_0,\mathbf{a}(1)\mathbf{a}(2)\mathbf{a}(3)\mathbf{a}(4)\mathbf{a}(5),5)$ on Figure 4 or the trajectory for the Blue bomber UAV to attack the Red radar (from (5,5) to (7,8) on Figure 5.
- *1st negation trajectories*, each representing a path for an agent of an opponent of the main player to hinder the advancement of the agent of the main player along the main trajectory. E.g., $\mathbf{t}(q_3,\mathbf{a}(6)\mathbf{a}(7)\mathbf{a}(4),4)$ and $\mathbf{t}(q_2,\mathbf{a}(8)\mathbf{a}(9)\mathbf{a}(4),4)$ on Figure 4 or the trajectory for the Red Fighter to attack the Blue Bomber (from (1,1) to (5,5) on Figure 5.
- *2nd negation trajectories*, each representing a path for an agent of the main player to hinder the advancement of an agent of an opposing player along a 1st negation trajectory. E.g., $\mathbf{t}(p_1,\mathbf{a}(13)\mathbf{a}(9),3)$ **and** $\mathbf{t}(p_2,\mathbf{a}(10)\mathbf{a}(12),2)$ on Figure 4 or the trajectory for the Blue Fighter to attack the Red Fighter (from (13,9) to (9,5) on Figure 5.
- *3rd negation trajectories*, ...

...

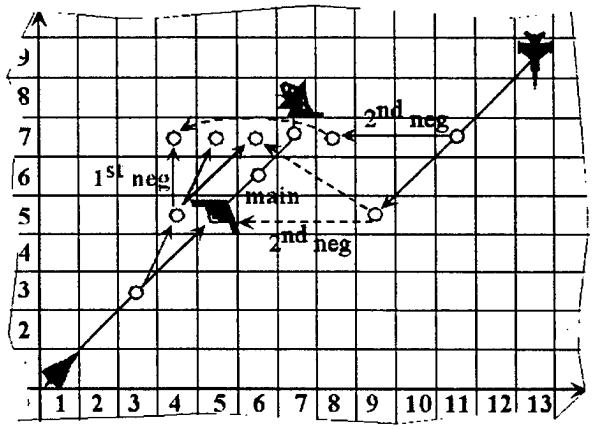


Figure 5. A Zone from the **4AFBE** example

The set of various other networks is represented in LG as a set of Languages of Webs, Zones, etc. [9].

3.2.3 LG Search Trees

To generate a move in the Start State of a Complex System, we initiate the search procedure. This procedure is not a usual search over a data structure but the construction of optimal (sub-optimal) variants of transitions leading from the Start State to the target states. These variants will reflect the best LG strategy for one side assuming that the opposing side follows its own best LG strategy at every state. This construction does not happen as a one-time action. It begins at the Start State, then the system moves to a new state while pursuing one of the *local goals*. The set of local goals is generated employing trajectories and networks. At the new state, a set of local goals is re-constructed, which leads to the third state, and so on. While these local goals are certain sets of states of a Complex System, they are

described in terms of *desirable events* with respect to the elements of subsystems and their locations. For example, element p_0 should reach the specific location and/or destroy the opposing element q_4. Another element q_1 should prevent this event by intercepting p_0, etc. These descriptions are formalized in the Language of Zones (see above). Various types of Zones reflect various types of local goals, while the trajectories within a Zone reflect the set of paths for a group of elements to pursue these local goals.

The result of the above search procedure is represented via another LG tool called the LG Search Tree. Those trees, see Figure 6, represent local optimal strategies for both players. The final strategy is represented by generation of successive LG search trees.

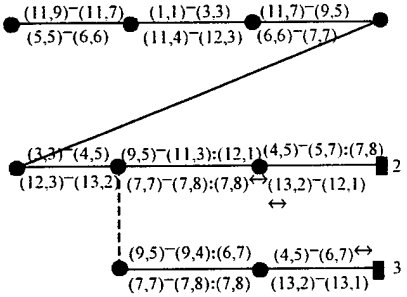


Figure 6. The LG search tree for the **4AFBE** example and a supplemental non-optimal branch

In the case of totally concurrent games (see Section 3.2.4), every node of the LG search tree (see Figure 6 and Section 4) represents a concurrent move involving all the components. Three kinds of component moves are employed:

- normal, e.g., (13,9)-(11,7) for the Blue Fighter;
- eliminating, e.g., (9,5)-(11,3):(12,1) means that the Blue fighter moves and eliminate the Red Bomber at (12,1);
- terminal, e.g., (4,5)-(6,7)× means that the Blue fighter moves and is destroyed.

It is not necessary to actually reach the local goals for both sides. One of the opposing sides will usually prevent this mutual success by reaching its local goal first, e.g., by protecting the target element successfully. However, new local goals may appear within the horizon, and the pursuit continues. It is our *assumption* that by moving from one local goal to another (or just by pursuing them) we will eventually reach the global goal, and this is going to be a satisfactory or, perhaps, an optimal solution. The optimality holds for the class of Reti-like (RL) problems (Section 3.2.4). It is likely that the result of optimality can be generalized beyond the class of RL problems. In the meantime, our assumption is supported by the fact that highly-skilled experts in various problem domains (including chess) evaluate the LG solutions as solutions of high quality, and they are precisely the *best known solutions*.

3.2.4 The LG Quality Function

The LG approach avoids extensive search by outright rejection of trajectories and zones of inferior quality. The quality function measures several characteristics, including the following:

- Passability (of trajectories and zones)
  - *LG internal mechanisms:*
    - obstacles for trajectories, either physical or passing through the weapons range of an adversarial agent. If those are present, the respective trajectory is impassable.
    - final material loss for zones. If executing an attack zone would cause predetermined material loss, the zone would be considered impassable.
  - *External mechanisms:*
    - Another control theory method may determine that an LG trajectory is not achievable
    - Intelligence reports that the enemy would not undertake certain actions
- Length (for trajectories).
- Material index (trajectories and zones).
- Sharing Zones (trajectories). If a trajectory is in more than one zone, its quality significantly increase.
- Coverage (trajectories and zones). If a trajectory contains a location such that all of its possible exits are within the weapons range of a potentially good enemy move, its quality decreases.

Finally the quality function is recursive and is computed during several iterations permitting to reject increasing number of trajectories and zones.

3.3 RL Problems

In various publications listed in (Stilman,2000), we considered a series of examples of combat simulation problems. These were various generalizations of the R. Reti chess endgame (1920), specifically, the problems with 4 mobile agents for 2D/3D operational district and various types of serial/concurrent combat. All of them and many different, significantly more complex, problems have been solved employing the LG strategies [9]. What is particularly interesting about this series of the 4-agent problems is that they can be unified as one

class of problems, the Reti-like (RL) problems.

DEFINITION 3. *Reti-like problems* (RL) is the class of Complex Systems (abstract board games) that is defined as follows.

The operational district X is a d-dimensional (dD) cube with each side of size n. The district includes n^d locations. Every location is given by the d-vector of integer coordinates $x = (x_1, x_2, ..., x_d)$, $x \in \mathbf{Z}^d$.

Set P includes four mobile agents that are organized in two opposing sides, P_1 and P_2, (which we will verbalize as Black and White). Each team includes a STATION and an INTERCEPTOR; for problems with 2D operational district they are called a BOMBER and a FIGHTER, respectively.

A STATION can move straight ahead towards the goal area, one location at a time. The goal location is within the cube X. Thus, the movements of a STATION can be reflected by incrementing (by 1) one of its coordinates, e.g., x_s, x_s+1,x_s+2, etc., and leaving the rest of the coordinates unchanged. After reaching its goal, a STATION becomes immobile. The goal location is the safe location for the respective STATION: if it reaches this location and stays there for at least one time interval, it cannot be destroyed. A STATION has weapons capable of destroying the opposing INTERCEPTOR at the forward adjacent diagonal locations. To formally define this set, let us introduce the set of locations adjacent to location x, ADJACENT(x). This set of locations can be described by incrementing (by 1) an arbitrary number of coordinates (at least one) of $x = (x_1, x_2, ..., x_d)$ and leaving the rest of the coordinates unchanged, i.e.,

$$\text{ADJACENT}(x) = \{y \mid y = (y_1, y_2, ..., y_d) \wedge x \neq y \wedge \forall i \in [1..d]\ (y_i = x_i \vee |y_i - x_i| = 1)\}$$

Thus, a STATION located at x can destroy the opposing INTERCEPTOR at the locations from ADJACENT(x).

An INTERCEPTOR can move to any adjacent location within the cube X. If an INTERCEPTOR is at location x then the relation of reachability $R_{\text{INTERCEPTOR}}(x, y)$ holds if $y \in$ ADJACENT(x). An INTERCEPTOR is able to destroy an opposing STATION by approaching its location from any direction and moving to this location. Then, the STATION is removed from the Complex System during the same time interval. If a STATION comes to the location, where the opposing INTERCEPTOR is, it is destroyed and removed from the Complex System as well. An INTERCEPTOR is capable of protecting its friendly STATION. In this case, the joint protective power of the combined weapons of the friendly STATION and the INTERCEPTOR (from any location adjacent to the STATION) can protect the STATION from interception.

Let us identify three subclasses of problems with respect to the type of the Complex System. They are alternating serial (AS), alternating concurrent (AC), and totally concurrent (TC). The rest of the definition is different for each of the subclasses.

Alternating serial (ASRL) problems include problems in which only one agent at a time can move; also, White and Black alternate turns. Additionally, for the AS problems the following goals hold. The goal of each side is to reach the set of winning target states where the opposing STATION is destroyed and the friendly STATION reached its goal location and has been safe for at least one time interval thereafter. If this is impossible for a given side, then its goal would be to reach the draw target states, where both STATIONs reached their goal locations and stay safe for at least one time interval, or where both STATIONs are destroyed before they reach their targets or immediately thereafter.

Alternating concurrent (ACRL) problems include problems in which agents of the same side can move simultaneously, while White and Black alternate turns. Additionally, for AC problems the following conditions and goals hold. A STATION is vulnerable not only to the opposing INTERCEPTOR attack but also to the explosion of another STATION. If an INTERCEPTOR would hit the opposing STATION while the latter is not at its goal location, and if the friendly STATION is moving during the hit, it would be destroyed as well as a result of the opposing STATION explosion. If the friendly STATION is not moving at this moment, it would be safe. The goals for each side are the same as for the serial problems.

Totally concurrent (TCRL) problems include problems in which all the agents can move simultaneously. Additionally, for TC problems the following conditions and goals hold. A STATION cannot move and destroy the opposing INTERCEPTOR at the forward adjacent diagonal locations. Additional vulnerability of the STATIONs is the same as for the ASRL problems. The goal of each side is to reach the set of winning target states where the opposing STATION is destroyed and the friendly STATION reached safely its goal location. If this is impossible for a given side, then its goal is to reach the draw target states where both STATIONs reached safely their goal locations or both STATIONs are destroyed before they reach their goalsor at their destinations. TCRL problems are the models of abstract board games with *incomplete information* (about the current move). Therefore, a deterministic strategy may not exist; the actual strategy can be implemented with certain probability (Stilman,2000).

THEOREM 1. A Reti-like problem with an arbitrary Start State, dimension d, and size n of the operational district can be solved by construction of an LG strategy (without a game tree search). This strategy is an optimal solution. The LG algorithm runs in polynomial time, $O(n^{3d+2})$, on the class of RL problems.

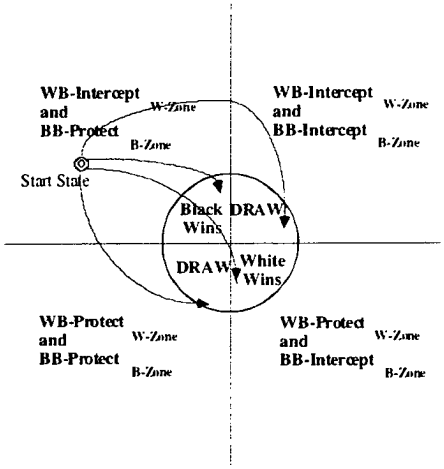


Figure 7. The State Space Chart

The proof consists of two parts. The first part includes a formal description of the full set of strategies which begins with the constructive expansion of the terminal sets employing Languages of Trajectories and Zones. This expansion allows us to realize a decomposition of the State Space in the form of a State Space Chart (see Figure 7). The second part of the proof includes an outline of the strategy-candidates following the guidelines of the State Space Chart (shown by arrows in Figure 7) and construction of one of them. The construction is accompanied by the proof of correctness of the solution.

An approach based on the conventional search algorithms, e.g., mini-max search algorithm with alpha-beta pruning, is intractable because it generates exponential searches. It runs at best in $O(3^{d*n})$ time [9].

An example of the serial ASRL problem for an arbitrary size 3D district is shown in Fig. 4.2. Let us introduce an $n\times n\times n$ district with B-STATION at $(4,1,n-2)$. It is shown in Fig. 4.2 as a double black ring. The W-STATION is shown as a double white ring. The goal locations for the W- and B-STATIONs, on the locations $(2,6,n)$ and $(n,1,n-2)$, respectively, are shown as gray cubes. The INTERCEPTORs and their locations are also shown in Fig. 4.2. A strategy generated by the LG algorithm for this problem forces draw for White. The key is for W-INTERCEPTOR to move around the main diagonal of the cube and at some point of time to deviate from it either towards the W-STATION or towards the B-STATION.

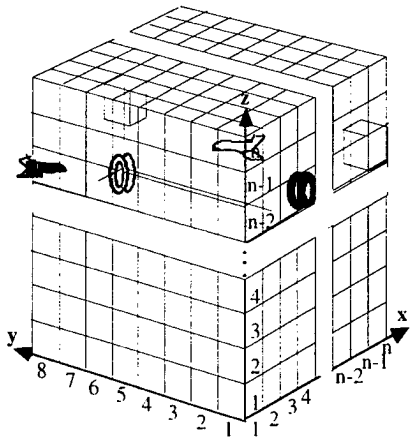


Figure 8. The 3D/4A AS RL problem for an $n\times n\times n$ district

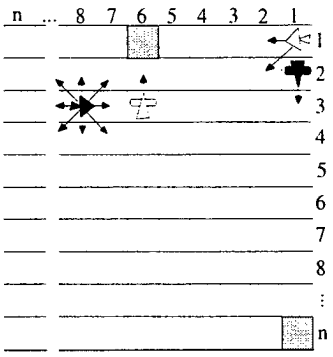


Figure 9. The 2D/4A TCRL problem for an n×n district

An example of the TCRL problem is shown in Figure 9. The operational district X is a 2D $n\times n$ square grid, $n>7$. The W-FIGHTER located at (1, 1), can move to any adjacent square. It can reach any of the points $y \in \{(1, 2), (2, 2), (2, 1)\}$ in one step, i.e., the reachability $R_{W\text{-}FIGHTER}((1, 1), y)$ holds. The B-BOMBER from (1, 2) can move only straight ahead, one square at a time, e.g., from (1, 2) to (1, 3), from (1, 3) to (1, 4), etc. The B-FIGHTER located at (8, 3) can move to any adjacent square. The W-BOMBER located at (6, 3) is analogous with the B-BOMBER, i.e., it can move only straight ahead but in opposite direction. It can reach only (6, 2) in one step. A strategy generated by the LG algorithm for this problem is a draw strategy. The W-INTERCEPTOR should move diagonally to (2,2), (3,3), etc., and at some point of time deviate towards the W-BOMBER or the B-BOMBER. The specific deviation depends on the movement of Black's agents.

3.4 Difficulties within the Problem Domains and some LG Techniques to Overcome them

- Difficulty: Hard to compare possible variants:

- *Solution: The LG Zone "quality" function provides smooth comparison*

- Difficulty: Uncertain, to which depth a variant should be computed:
  - *Solution: Two LG mechanisms:*
    * "Rolling" visibility horizon for agents
    * variant extrapolated outcome function
- Difficulty: Enormous number of variants of possible behavior due to large branching factors. E.g., computation of variants of depth 3 on the infinite board for the 4AFBE example (see Section 4) will give us $(9^2 * 29^2)^3 \approx 3 * 10^{14}$ variants.:
  - *Solution: LG Zones:*
    * structurally limit variants of possible confrontation
    * only highest "quality" Zones are utilized in strategy generation

4. EXAMPLE: "FOUR AGENT FIGHTER-BOMBER ENGAGEMENT" (4AFBE)

4.1 Problem Settings

4.1.1 Initial Game Board State

In our example, there are two players, the Blues and the Reds. Each player has a fighter, a UAV bomber, and a ground installation. The original disposition of forces is given on Figure 10. We will consider a totally concurrent game (see DEFINITION 3), i.e., all possible concurrent moves are allowed.

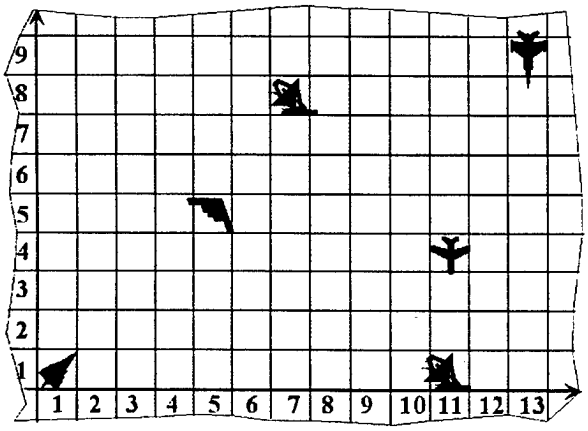


Figure 10. An abstract board for **4AFBE**

4.1.2 Deriving the Winning Condition from the Mission Objectives

We can describe the mission objective for the Blues (see Figure 10) as a list of prioritized goals:

- *Priority 1 Protect the blue radar (G1);*
- *Priority 2 Destroy the red radar (G2);*
- *Priority 3 Protect the blue bomber or destroy the red fighter (G3).*

This list yields the following evaluation function (also known as a pay-off function) ***Eval***:

- ***0** if not G1*
- ***1** if G1 and not (G2 or G3)*
- ***2** if G1 and G2 and not G3*
- ***3** if G1 and G2 and G3*

In general, we do not limit ourselves to zero-sum games and thus the Reds may have a list of goals not necessary symmetric to the one for the Blues. However, for the simplicity sake, we will assume that all the Reds want is to hinder the Blues as much as they can. Thus, in terms of the abstract board games, the adversaries have the following conflicting goals:

- *Blues: maximize **Eval***
- *Reds: minimize **Eval***

4.1.3 The Game Rules

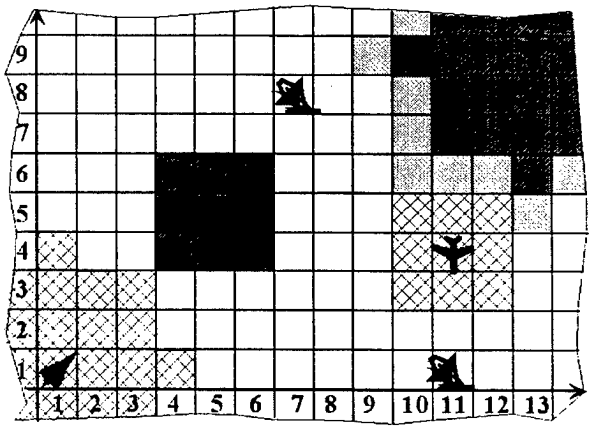


Figure 11. Reachabilities and weapons ranges of agents

Finally, in order to complete 4AFBE we need to describe how to move agents and their weapons capabilities, (also see Figure 11):

- Each bomber's weapons range is its current location
- Blues:
  - *Dark squares describe range of movements in one step*
  - *Lighter gray squares describe the weapons range for the fighter*

- Reds:
  - *Checkered squares describe range of movements in one step*
  - *The fighter have the same weapons range as its movements*

4.2 An Intuition-Based Solution without LG

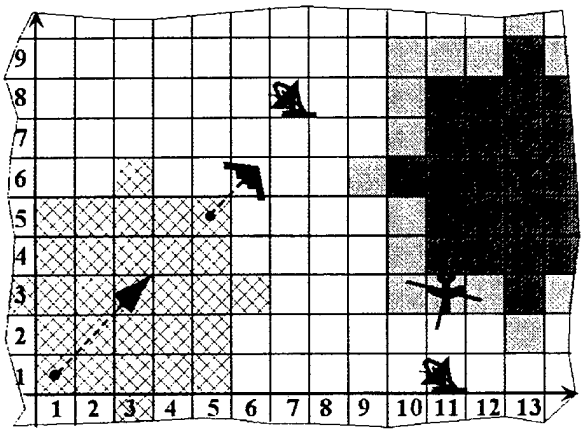


Figure 12. Intuitive move 1

An intuitive attempt to maximize ***Eval*** is described on Figure 12 and Figure 13. In this simplified scenario, if an adversarial agent is within the weapons range of a piece, its destruction is immediate, i.e., without wasting a move. In general, other interpretations of the weapons range are possible, i.e., launching a missile that would be represented as another piece. In such case, the destruction of the adversarial agent may not be imminent.

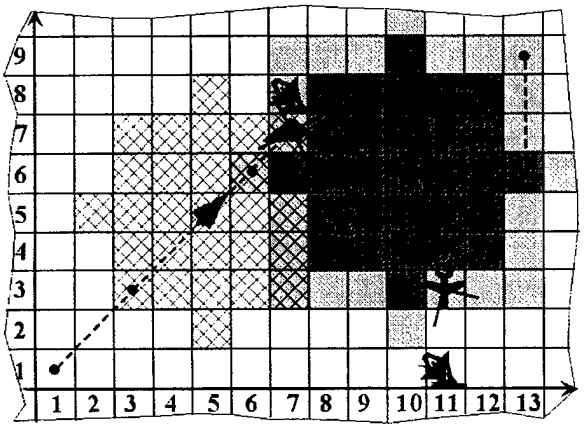


Figure 13. Intuitive move 2

The intuitive behavior results in ***Eval*** = 1. However, there is a solution with ***Eval*** = 2, and we'll apply the LG strategies to find it.

4.3 The LG Solution for 4AFBE

4.3.1 Generating highest quality Zones

From the LG point of view, this problem is simple because all major Zones and trajectories can be generated in the Start State. Two shortest main trajectories of the Blue Bomber leading from (5,5) to (7,8) are shown in Figure 5 and Figure 16, providing the basis for two highest quality zones associated with the Blue Bomber. There are two 1st negation trajectories of the Red Fighter from (1, 1):

$a(1,1)a(3,3)a(4,5)a(6,7)$ and $a(1,1)a(3,3)a(5,5)a(7,7)$.

Existence of at least one such a trajectory means that the Red Fighter has enough time to intercept the Blue Bomber at (6, 7) or (7, 7), respectively. Recall that the agents move simultaneously. Also, an interception of the Blue Bomber is possible at the destination (7, 8), where the Blue Bomber will be destroyed after it already hit the Radar. However, the 1st negation trajectory $a(1,1)a(3,3)a(5,5)a(7,7)$ is not really safe for the Red Fighter. Indeed, the 2nd negation trajectory of the Blue Fighter from (13, 9):

$$a(13, 9)a(11, 7)a(9, 5)a(5,5)$$

shown in Figure 5 and Figure 16 makes it unsafe (formally, of inferior quality, see Section 3.2.4). Thus, the Red Fighter must choose another intercepting trajectory $a(1,1)a(3,3)a(4,5)a(5,7)a(7,8)$ which would be the highest quality 1st negation trajectory.

All the shortest main trajectories of the Red Bomber are shown on Figure 16. Each is of the length 3. The Red Bomber will be intercepted by the Blue Fighter moving along one of the 1st negation trajectories that include a common path $a(13,9)a(11,7)a(9,5)$. The weapons range of the Blue Fighter applied from location (9,5) shows that none of the shortest main trajectories of the Red Bomber are safe because the Blue Fighter "dominates" all the locations (see Figure 14). The only way for the Red Bomber to approach the Radar at (11,1) is to try to use a longer trajectory. Let us generate all the main trajectories of the length 4. They are shown on Figure 15. Again, taking into account the weapons range of the Blue Fighter applied from (9,5), we conclude that the only main trajectory of the Red Bomber that might be safe from the Blue Fighter after the 1st move is $a(11,4)a(12,3)a(13,2)a(12,1)a(11,1)$. This is the highest quality main trajectory. Notice that the sub-trajectory $a(12,3)a(13,2)a(12,1)a(11,1)$ is safe because there are no intercepting trajectories of the Blue Fighter from (13, 9) leading to this location: it does not have enough time (one time interval).

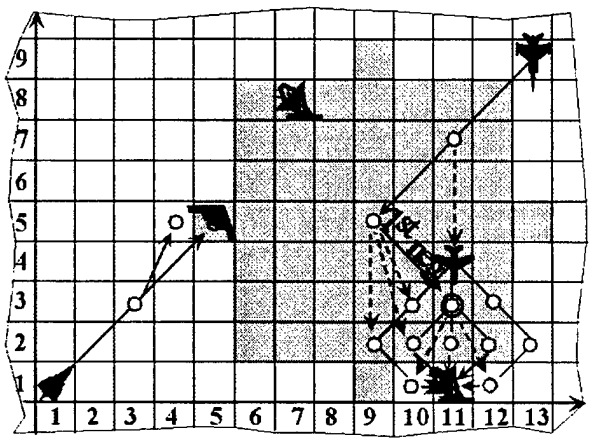


Figure 14. Domination of Blue Fighter

Finally, out of all the numerous intercepting trajectories of the Blue Fighter from (13,9) leading to the locations of the main trajectories of the Red Bomber (Figure 14 and Figure 15) and to the locations of the 1st negation trajectories of the Red Fighter (Figure 5), the only bundle of trajectories with the common path ***a***(13,9)***a***(11,7)***a***(9,5) can serve both goals. By moving along these trajectories the Blue Fighter will achieve the following:

- force the Red Bomber to follow the longer, round about route and lose time on its way to the Blue Radar;
- force the Red Fighter away from the intercepting trajectory leading to (7,7) (Figure 14) and allow the Blue Bomber to hit the Red Radar.

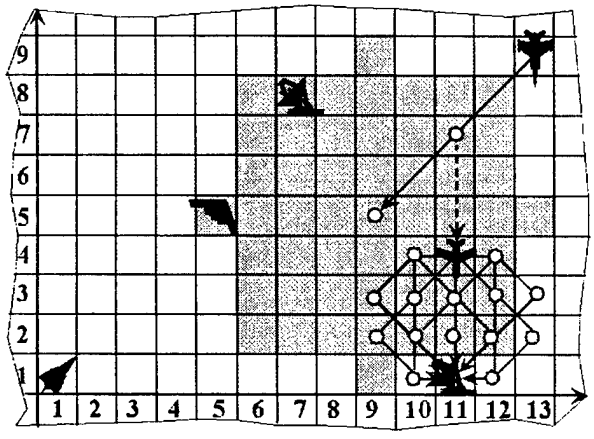


Figure 15. All the main Red Bomber trajectories of length 4

As a first step to produce a formal LG-based solution we would have to generate a number of Zones associated with the initial board. They would be generated by the LG Grammar of Zones. As was indicated before, see Section 3.2.4 a great number of them would be rejected due to inferior quality. In the discussion above, we informally described how the quality function would identify zones of inferior quality. The highest quality Zones for **4AFBE** are depicted on Figure 16.

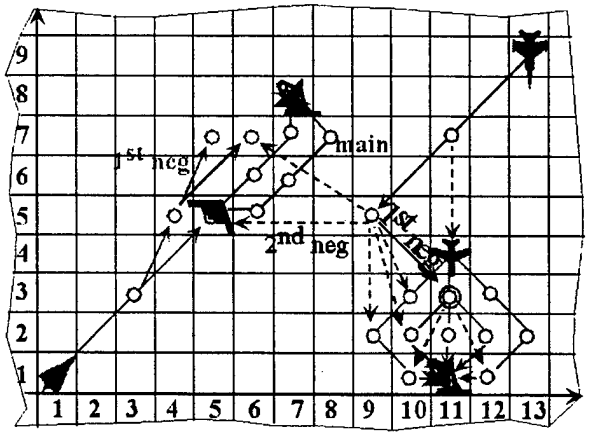


Figure 16. Highest quality Zones for **4AFBE**

4.3.2 The LG Search Tree and its Implementation

The above comments are intended to give a general idea how the LG algorithm solves the problem and generates the LG tree shown in Figure 6. This tree includes only one optimal branch and a supplemental non-optimal branch.

The LG search tree is different from the conventional search trees. Every concurrent 2-side move is represented by *two consecutive arcs* - two components. The arc outgoing from the blue node represents the Blue's component of a concurrent move, while the arc outgoing from the red node represents the Red's component of the *same* move.

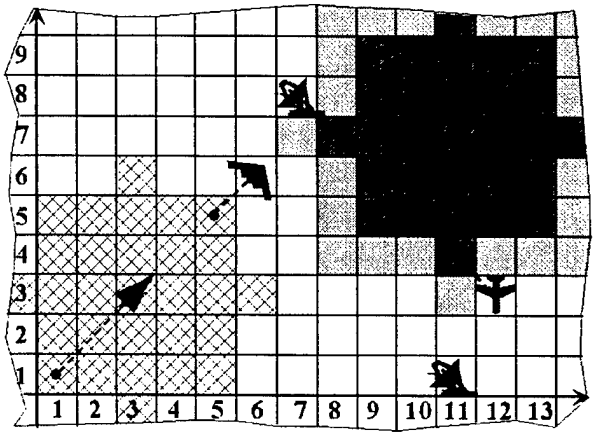


Figure 17. The first move

The LG search tree generated from the initial Zones is described on Figure 6. It represents a solution for the

example. It may be shown, that no matter what the opponent do, the Blues may achieve ***Eval*** = 2. We added to Figure 6 a supplemental tree that shows non-optimal moves for the Reds resulting in ***Eval*** = 3.

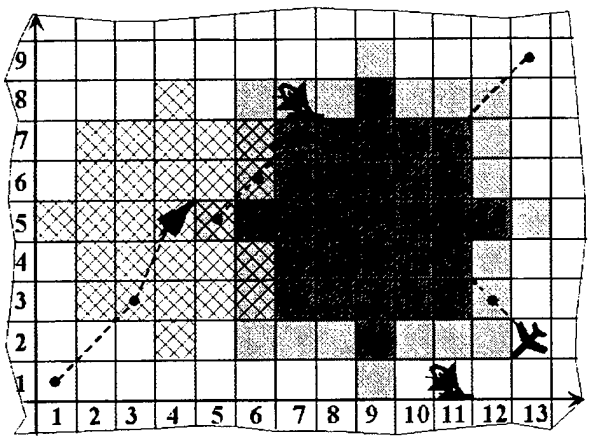


Figure 18. The second move

The implementation of the LG search tree as actual game moves is illustrated on Figure 17, Figure 18, and Figure 19.

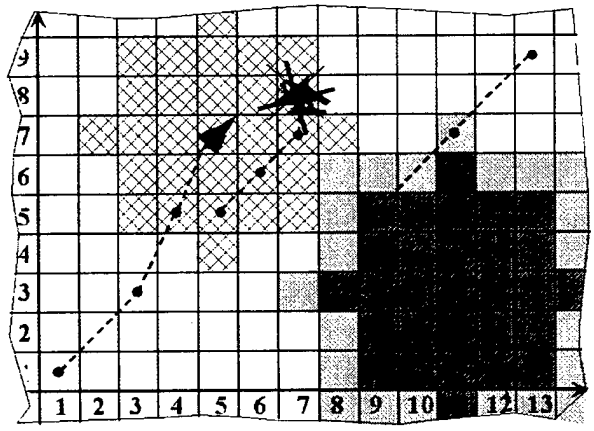


Figure 19. The third move

5. CONCLUSION

Armed with an LG strategy, battlefield commanders may choose behaviors that will eventually bring about a realization of their shared and/or separate goals:

- *a group of UAVs sharing a goal may be totally controlled by a strategy*
- *a squadron of manned aircraft may be provided with an advice that can be either followed, modified, or overridden*
- *a loose congregation of battlefield units may better coordinate their behavior*

Similar techniques may be applied to strategic problems where the component agents are wings and squadrons, instead of single aircraft.

References

1 Fikes, R.E. and Nilsson, N.J. (1971). STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving, *Artificial Intelligence* 2, (189–208).

2 Knuth, D.E., Moore, R.W., (1975). An Analysis of Alpha-Beta Pruning, Artificial Intelligence, (293-326), 6(4).

3 Nilsson, N.J. (1980). *Principles of Artificial Intelligence*, Palo Alto, CA: Tioga Publ.

4 Sacerdoti, E.D. (1975). The Nonlinear Nature of Plans, *Proc. Int. Joint Conference on Artificial Intelligence*(206-214).

5 Stilman, B., (1994d). A Linguistic Geometry for Control Systems Design, *Int. J. of Computers and Their Applications*, (89-110), Vol. 1, No. 2, Dec. 1994.

6 Stilman, B., (1997a). Managing Search Complexity in Linguistic Geometry, IEEE *Transactions on Systems, Man, and Cybernetics*, Vol. 27, No. 6, pp. 978-998, Dec. 1997.

7 Stilman, B., (1997b). Network Languages for Concurrent Multiagent Systems, *An International Journal: Computers & Mathematics with Applications.*, Vol.34, No. 1, pp. 103-136, 1997.

8 Stilman, B., Fletcher, C., (1998). Systems Modeling in Linguistic Geometry: Natural and Artificial Conflicts, *International Journal: Systems Analysis, Modeling, Simulation*, (57-97), Vol. 33, 1998.

9 Stilman, B. (2000). *Linguistic Geometry: From Search to Construction*, 404 pp., Kluwer Acad. Publishers, Jan. of 2000.

10 Yakhnis, A., Yakhnis, V. (1993) Gurevich-Harrington's Games Defined by Finite Automata, *Annals of Pure and Applied Logic*, Vol. 62, pp. 265-294, 1993.

11 Yakhnis, V., Stilman, B., (1995) A Multi-Agent Graph-Game Approach to Theoretical Foundations of Linguistic Geometry, *Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence* (WOCFAI 95), Paris, France, July 1995.

12 Zeitman, S. (1994) Unforgettable forgetful determinacy, *Logic and Computation*, Vol. 4, pp. 273-283, 1994.

A Constructivist Theory of Distributed Intelligent Control of Complex Dynamic Systems

Dr. Shashi Phoha, Dr. Eileen Peluso, Dr. Richard Brooks
Information Science and Technology Division
Applied Research Laboratory
The Pennsylvania State University
P. O. Box 30
State College, PA 16804-0030
Tel. (814) 863-1131
Fax. (814) 863-1396
Email: sxp26@psu.edu

Abstract

This paper develops a constructivist theory of distributed, intelligent command and control for collaborative and/or hostile autonomous agents in a dynamic and uncertain environment. The theory is constructivist in the sense that it constructs a process algebra of agent behaviors for dynamic task decomposition and iterative control synthesis in a hierarchical command and control structure. Agents self-organize in multi-layered C^2 subnetworks of a connected C^2 network to execute a mission. The top-level mission script is decomposed into sub-scripts for the next lower levels much as a higher order language instruction is decomposed into machine instructions. Execution is monitored by the higher levels through observation of events generated or reported by lower level C^2 agents. A supervisory control function ϕ enables or disables behaviors of lower level agents to enable or disable a specific pattern of controllable events, thus establishing an iterative control loop at each level of the C^2 hierarchy. The paper formulates distributed command and control of autonomous agents as follows: (i) the forward problem of predicting emerging global behavior for observations of local agent behaviors and (ii) the backward problem of achievement of desired global behavior of the system through control synthesis and coordinated behaviors of individual autonomous agents.

1. Introduction

The dynamic time evolution of the complex interactions of processes in command and control (C^2) systems is inherently different from that modeled in classical control theory using differential or difference equations. While the classical theory attempts to characterize continuous variable dynamics of physical processes, command and control processes are triggered by discrete events and are characterized by a large number of interacting operations under predictable and unpredictable disturbances. Unlike the evolution of physical processes, there are no inherent physical laws to constrain system configurations other than natural limitations of human comprehension, resources, and ergonomics. Hence, combinatorial explosion of complexity is characteristic of such systems. Consequently, successful, performance-oriented models of C^2 systems must dynamically constrain computational requirements to "good enough" feasible solutions which can be computed under dynamic time constraints and available resources.

The analytical formulation of multistage control strategies for a network of autonomous agents is modeled mathematically in this paper as a dynamically self-modifying algorithm [8, 16] for the cellular space generated by agent interactions with each other and with the environment (figure 1). Following the pioneering work of Ramadge and Wonham [17], C^2 agents have been modeled as supervisory controllers of discrete event dynamic systems [14, 15]. In [14, 15], C^2 of military

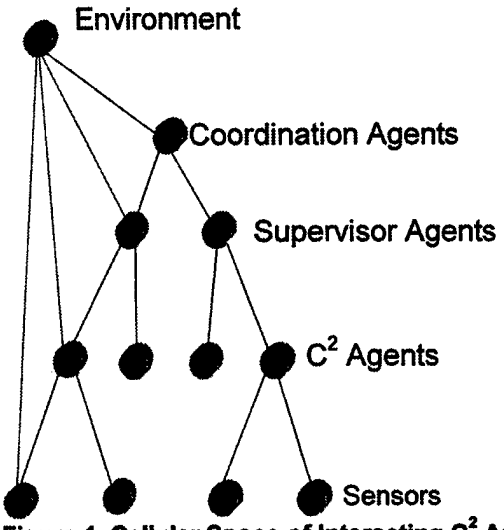


Figure 1. Cellular Space of Interacting C^2 Agents

forces was first formulated as the design of a supervisory controller for a set of hierarchically interacting automata. At the same time, extensions of these models to dynamically reconfigurable and hierarchically controlled organizations for slowing down the combinatorial explosion were also explored at Penn State University [4, 13]. The intermediate level intelligent controllers were modeled as transducers with the dual roles of supervisory control for the assigned lower layers and generator of the dynamics for a higher layer which provides feedback control [12]. Multistage controllability is mathematically defined in this work by extending controllability definitions of Ramadge and Wonham. The present work builds a unified computational theory of distributed, intelligent C^2 of heterogeneous and possibly inconsistent autonomous agents (friendly or enemy) by enhancing previous work with the following innovations:

- A process algebra is introduced for behavior composition
- A general behavior message passing language is introduced for modeling concurrency and distributed agent interaction,
- Controllability, stability and observability of the system are defined, and
- Multi-layered controllability is formulated in terms of iterative controllability at adjacent layers of autonomous agents.

Two fundamental problems are mathematically formulated in this context as follows:

The *forward problem* is that of predicting emerging global behavior from observations of local events which cause state transitions for individual autonomous agents.

The *backward* or *control synthesis problem* is that of dynamic mission decomposition and iterative behavior control of a group of autonomous agents to achieve desired global behavior.

These problems are approached using resource-bounded computation to build flexible and intelligent systems operating in dynamic and uncertain environments. The approach is also known as flexible optimization [8], anytime algorithms [2], imprecise computation [10], and design-to-time [5]. The objective is to tradeoff global optimality of solution with the resources (including time and memory requirements) available for reaching and executing a solution.

Section 2 of this paper presents the mathematical formulation of control laws as a supervisor of interacting autonomous agents which restricts the system to desired behaviors. Section 3 models concurrency and distributed interactions of agents to analytically formulate the control analysis and synthesis problems. Stability, controllability and observability are formally defined. Section 4 introduces a process algebra for behavior composition facilitating the process by which complex compositions of elementary behaviors of agents can be recognized and controlled by a supervisor at an aggregate level. Notions of behavioral equivalence are formalized to support supervisory verification. Finally section 5 presents necessary and sufficient conditions for controllability and completeness of supervisors in a multi-layered control system.

2. Mathematical Formulation of Multi-Layered Hierarchical Control Problems

A distributed collection of autonomous discrete event systems, or *agents*, under hierarchical control is modeled by a tree of interacting automata. The formalisms for the automata used in the hierarchical automata-based control model (figure 2) are well established in the area of formal language and automata theory [6]. The lowest level agents (leaf nodes) are modeled by *generating automata*, whose alphabets represent discrete events. Note that it is not necessary that those leaf nodes be at the same depth, as shown in figure 2. As the agents operate, discrete events are generated representing the behavior of the agent over time. Certain of these events are *controllable* in that their occurrence can be influenced by outside forces. The *language*, defined as the set of all possible sequences, or *strings*, of output symbols, of a leaf node models the *open-loop behavior* of that node, i.e., the behavior of the node when no outside forces are in place to influence its behavior.

Internal nodes in the hierarchy are modeled as *transducers* that observe the events generated by the next lower level and respond by *enabling* and *disabling* controllable events. In this fashion, the higher level nodes exert control over subordinate nodes, restricting their behavior to desirable subsets of the open-loop behavior. While exerting control over subordinates, these transducers recognize strings of events and report their occurrences as single symbols of a *higher level alphabet* to the next higher level in the hierarchy. This higher level alphabet is therefore a coarser representation of the behavior of the system. Control is exerted by the higher level nodes in the same fashion, by enabling and disabling higher level controllable events. Although the high level nodes only directly control their immediate subordinates, the effects of this control are manifested at all appropriate subordinate levels within the hierarchy.

This model provides a formal language representation of the operation of a hierarchical structure of autonomous agents. The behavior of the uncontrolled system is represented by the *language* or set of all possible strings of events generated by the leaf nodes in the hierarchy. By imposing an appropriate set of supervisors, the behavior of the system is restricted to that subset of strings that represents desired or appropriate system behaviors.

2.1 Mathematical model

We model each autonomous agent as an automaton using standard formalisms [6] as follows:

$R = (Q, \Sigma, \delta, q_0, Q_m)$, where

- Q is a set of states, possibly infinite,
- Σ is a finite alphabet consisting of discrete events which cause state transitions,
- δ: $Q \times \Sigma \rightarrow Q$ is a partial function denoting state transitions,
- $q_0 \in Q$ is the initial or start state, and
- $Q_m \subseteq Q$ is a set of marked states.

Let Σ^* denote the set of all finite strings of elements of Σ, including the empty string ε. Any subset K of Σ^* is called a *language*, and its closure is the set

$$\bar{K} = \{s \in \Sigma^* \mid \exists t \in \Sigma^*, \text{ such that } st \in \bar{K}\}$$

The function δ is extended to strings in K in the usual fashion. The fact that $\delta(q,s)$ is defined is denoted by $\delta(q,s)!$ and the following languages are defined:

$$L(R) = \{w \mid w \in \Sigma^* \text{ and } \delta(q_0, w)!\}$$

$$L_m(R) = \{w \mid w \in L(R) \text{ and } \delta(q_0, w) \in Q_m\}.$$

These languages can be viewed as being generated by R or as being recognized by R.

The intermediate level supervisors have a dual function of recognizer/generator, and to model this, we extend the automaton formalism to a *transducer* defined by a 6-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Q_m),$$

Where Q, q_o, and Q_m are as defined previously, and Σ is the input alphabet. Let Γ be the alphabet of symbols generated as output, and let δ map $Q \times \Sigma$ to $Q \times (\Gamma \cup \{\varepsilon\})$. When a transducer consumes an input symbol from alphabet Σ, it may or may not produce an output symbol from alphabet Γ. The transition function δ is extended to strings in the standard fashion.

We now give the necessary notation for each automaton in the hierarchy. Each generator R_i, $1 \le i \le n$, in Figure 2 is modeled as a generator

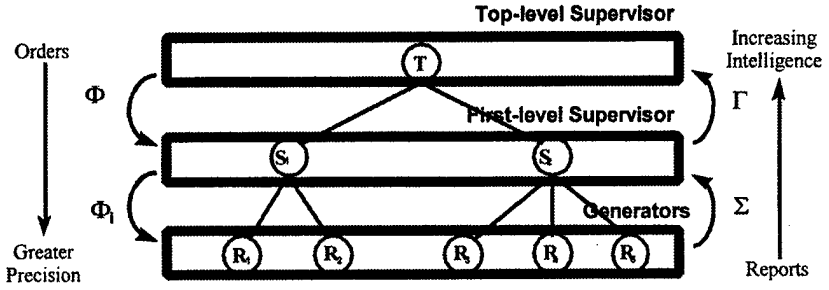
$$R_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, Q_{mi}),$$

with the restriction that the alphabets Σ_i, $1 \le i \le n$, be disjoint. Let $\Sigma = \cup \Sigma_i$.

The sets Σ_ϕ and Σ_u partition Σ into the subsets of controllable and uncontrollable events, respectively. Each supervisor S_i in figure 2 is represented by (S_i, ϕ_i), where $S_i = (X_i, \Sigma_i, \Gamma_i, \xi_i, x_{0i}, X_{mi})$ is a transducer, and $\phi_i : X_i \rightarrow 2^{\Sigma_i}$ is a state feedback mapping. The alphabets Γ_i, $1 \le i \le n$, are disjoint. Recall that for any state $x \in X_i$, $\phi_i(x)$ denotes the set of events in Σ_i that are enabled at state x and uncontrollable events are always considered to be enabled.

The top-level supervisor is modeled as $T = (T, \phi)$, where T is a recognizer: $T = (Z, \Gamma, \tau, z_0, Z_m)$ and $\phi : Z \rightarrow 2^{\Gamma}$, where $\Gamma = \bigcup_{1<i<n} \Gamma_i$.

The sets Γ_ϕ and Γ_u partition Γ into subsets of controllable and uncontrollable events, respectively. We let $\phi(z)$ denote the set of events in T's input alphabet, in this case Γ, that are enabled at state $z \in Z$. However, in



Generator $R_j = (Q_j, \Sigma_j, \delta_j, q_{0j}, Q_{mj})$, where Q_j is a set of states, Σ_j is the alphabet, δ_j is a partial mapping from $Q_j \times \Sigma_j$ to Q_j, q_{0j} is the starting state, and Q_{mj} is a set of marked, or final, states.
First-level Supervisor: $S_i = (S_i, \phi_i)$, where $S_i = (X_i, \Sigma_i, \Gamma_i, \xi_i, x_{0i}, X_{mi})$ is a transducer, where X_i is a set of states, Σ_i is the input alphabet, Γ_i is the output alphabet, ξ_i is a partial mapping from $X_i \times \Sigma_i$ to $X_i \times \Gamma_i \cup \{\varepsilon\}$, x_{0i} is the starting state, and X_{mi} is a set of marked, or final, states, and ϕ_i: $X_i \rightarrow 2^{\Sigma_i}$ is the state feedback mapping.
Top-level Supervisor $T = (T, \phi)$, where $T = (Z, \Gamma, \iota, z_0, Z_m)$ is a recognizer and $\phi : Z \rightarrow 2^{\Gamma}$, is the state feedback mapping.

Figure 2. Hierarchical Model of Interacting Interactive Automata

the hierarchical model, if $\gamma \in \Gamma_i$ is disabled, then additional symbols in Σ_i may also be disabled, depending on their relationship to γ. This effect of high-level control upon low-level events is described in detail in [13].

Conceptually, we see the automata interacting as follows:

- As a generator outputs a symbol, its associated first-level supervisor responds to the symbol by making a state transition and exerts control by enabling and disabling events in Σ.
- If that transition generates an output symbol from Γ, the top-level supervisor responds by making a state transition, thereby possibly enabling and disabling events in Γ.
- The generators are prohibited from generating symbols in Σ that are either *explicitly* or *implicitly* disabled. If a controllable symbol is disabled according to the feedback mapping of the first-level supervisor, it is said to be *explicitly* disabled. If, in the current state of the first-level supervisor, the input symbol σ is not explicitly disabled, yet the first output symbol generated on every trajectory beginning with σ is disabled by the top-level supervisor, the σ is *implicitly* disabled. In other words, a controllable symbol is implicitly disabled it if would necessarily lead to the generation of a string beginning with a disabled symbol in Γ.

We denote the languages generated under this hierarchical structure as *L(T/S/R)*. When this model is extended to hierarchies of four or more levels, all supervisors except the one at the top are modeled as transducers.

The ability of the hierarchical structure of supervisors to restrict the system to a specific language is dependent upon the aggregation present in the hierarchical structure. It is the transducers, the first-level supervisors, in the three-layer hierarchy that perform the aggregation of low-level events in high-level macro events. We model this aggregation as a partial mapping f: $\Sigma^* \to \Gamma^*$, where for $s \in \Sigma^*$, $f(s) \in \Gamma^*$ is the string generated by the first-level supervisors as they process the input string s.

Let *S/R* denote the n sets of interacting automata, and let $L(S/R) = \| L(S_i/R_i)$, for $1 \le i \le n$, i.e., the subset of Σ^* generated by the uncontrolled, asynchronous running of the n independent instances of generators and supervisors with no top-level supervision imposed. With the desired behavior of the system modeled by $K \subseteq L(S/R)$, we define controllability as the necessary and sufficient condition for the existence of a top-level supervisor T, such that $K = L(T/S/R)$.

3. Concurrency and Distributed Interaction of Multi-Agent Networks

In order to model the concurrent and interacting behaviors of autonomous agents in the complex dynamic system, we extend the automata representations of individual agents to form an interactive automata network, defined as a pair $(\mathcal{G}, \{\mathcal{A}_i\})$ consisting of a cellular space $\mathcal{G}$, a potentially countably infinite, locally-finite bi-directed graph, and an associated family of interacting automata $\mathcal{A}_i$, allocated to each vertex (cell) of a graph. In particular, $\mathcal{A}_0$ represents the operational environment. Each automaton $\mathcal{A}_i$ has a finite d_i number of neighbors to communicate with, and has the form

$\mathcal{A}_i = (Q_i, \Sigma_i, \Gamma_i, \delta_i, q_{0i}, Q_{Fi})$ where

Q_i is a finite set of control states representing values of all variables,

Σ_i is a finite alphabet of input events and $\Sigma_i = \Sigma_0 \times \Sigma_{i_1} \times \ldots \times \Sigma_{i_{d_i}}$,

Γ_i is a finite alphabet of output events and $\Gamma_i = \Gamma_0 \times \Gamma_{i_1} \times \cdots \times \Gamma_{i_{d_i}}$,

δ_i is a local transition function $\delta_i : Q_i \times \Sigma_i \to Q_i \times \Gamma_i$,

$q_{0i} \in Q_i$ is an initial control state, and

$Q_{Fi} \subseteq Q_i$ is the set of terminal control states.

The dynamic system operates locally as follows: an interactive automaton $\mathcal{A}_i$ occupies each vertex (cell) i of $\mathcal{G}$. Asynchronously, each $\mathcal{A}_i$ looks up its inputs from neighbors $x_{i_1}, \ldots, x_{i_{d_i}}$, input from an environment x_{i_0} and its own state x_i, and then changes its state and produces outputs for the neighbors and the environment according to a local dynamics δ_i (see figure 3). This atomic move is repeated any (possibly very large) number of times.

The environment is modeled uniformly as an interactive automaton, which can be nondeterministic or stochastic, assuming incomplete knowledge about the environment. A distributed environment is modeled as a subnetwork instead of a single node. Thus the local transitions δ_i induce a global evolution from one system configuration to another. This global evolution of the system is viewed as a self-map $\mathbf{T}: C \to C$, where $C = \prod_i (Q_i \times \Gamma_i)$ are configurations (total states) of the network, such that

$$\mathbf{T}(x)_i = \delta_i(x_i, x_{i_0}, x_{i_1}, \ldots, x_{i_{\delta_i}}).$$

The *orbit* of x is the sequence of configurations

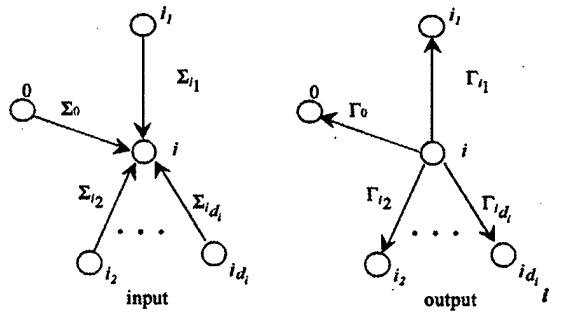


Figure 3. Input and output for transition rule δ_i, in neighborhood of node i.

$\{\mathbf{T}^t(x)\}_{t\geq 0} := x, \mathbf{T}(x), \mathbf{T}^2(x), \mathbf{T}^3(x), \ldots$
resulting from successive iterations of the global rule on x. As a dynamical system, the most basic question about a global map **T** is the effect of its repeated application in phase space to a random given configuration x.

3.1 The Control Analysis and Synthesis Problems

The *forward*, or *control analysis* problem is the following: given local transition rules that determine the local interaction of each automaton with its neighbors (namely the dynamics δ_i), characterize the global effect of the rules on an arbitrary initial configuration x. In other words, determine a specific description of the orbits of arbitrary configurations under T to identify their long-term (asymptotic) behavior. The forward problem is also known as the prediction of the emerging global behavior from local rules.

The synthesis problem is the inverse problem which, from the point of view of constructing control laws is even more important. Given a desired global effect on configurations, determine whether there exist, and if so, find the local rules δ_i whose induced global rule is precisely T. These local rules will then yield a parallel algorithm for the underlying parallel interacting automata to solve the problem of computing $T(x)$ for any configuration x in C.

3.2 Stability, Controllability and Observability in Interacting Automata Models

This section establishes some basic definitions of control characteristics of complex dynamic systems.

3.2.1 Network Stability

From classical control theory, we know that a continuous or discrete-time system is stable if every bounded input produces bounded output. Another equivalent definition assumes that a continuous system (discrete-time system) is stable if its impulse response $y\delta(t)$ (Kronecker delta response $y\delta(k)$) approaches zero as time approaches infinity. Intuitively, a stable system is one that remains at rest unless excited by an external source and returns to rest if all excitations are removed.

The network stability of an interactive agent system may take two forms: local or global. In local network stability we consider only a local control state of a single automaton; in global network stability we consider configurations (i.e., global states of the network).

We will consider strong and weak global network stability:

- in *strong network stability*, every finite configuration evolves to a stable configuration (a single set of terminal states) in finitely many steps;
- in *weak network stability*, every finite configuration evolves to an aperiodic configuration in finitely many steps.

In unstable interactive automata:

- with *weak network unstability*, an evolution on a random initial configuration leads to chaotic aperiodic patterns of configurations. However, an algorithm exists to decide whether one configuration belongs to the orbit of another;
- with *strong network unstability*, an evolution on a random initial configuration leads to complex patterns where no algorithm can be identified.

The definitions of local stability are analogous.

If we replace the finite sequence of global transitions by an infinite sequence, we will obtain the notion of *asymptotic stability*.

3.2.2 Controllability

The concept of controllability addresses the question of whether it is possible to control the state $x(t)$ (or sequence in the discrete case) from input $\sigma(t)$. Specifically, the controlled system is controllable if there exists a physically realizable sequence $\sigma_1; \sigma_2, \ldots, \sigma_n$ of inputs that will move the system from any point $x(t_0)$ in the state space to any other point $x(t_n)$.

DEFINITION 3.2.2 *The system is controllable if for a given initial configuration $x_0 \in C$ there exists a desirable terminal configuration $x_n \in C$, and a finite sequence of global transitions $T_1, \ldots, T_n$ such that*

$$(\bigcirc_{i=1}^{n} T_i)(x_0) = x_n,$$

where $\bigcirc$ denotes sequential composition.

3.2.3 Observability

The concept of observability is complementary to that of controllability. The controlled system is observable at t_0 if it is possible to determine the state $x(t_0)$ by measurement of the output y(t) (or output sequence in the discrete case) over a finite period of time.

DEFINITION 3.2.3 *An interactive automata network is observable if for the terminal configuration $x_n \in C$, and a finite sequence of global transitions $T_0, T_1, \ldots T_n$, it is possible to determine an initial configuration $x_0 \in C$, such that* $(\bigcirc_{i=1}^{n} T_i^{-1})(x_n) = x_0$.

4. Behavior Composition and Supervision

The major challenge now lies in the automated manipulation and intelligent coordination of causal behavior dependencies across multiple autonomous agents in the system. Process algebras, such as CCS [11], CSP [7], and ACP [1], adequately describe and analyze

interactive concurrently executing components of a dynamic system [9].

The output alphabet Γ_i of each autonomous agent $\mathcal{A}_i$ in the system consist of atomic, indivisible process steps called elementary behaviors. The set of elementary behaviors of an autonomous agent, along with a set of simple operations shown in Table 1, define a process algebra. More complex actions are expressed by combining elementary behaviors using the operations shown in Table 2. These operations define the process algebra known as \$-calculus [4].

The atomic process steps, shown in Table 1, capture elementary behaviors and are, for the most part, self-explanatory. Cost functions may be used for system adaptation and may consist of standard, probabilistic, and fuzzy cost functions. User defined cost metrics may be incorporated. Send and receive are used for handshaking message-passing communication, and also for inferencing. Crossover can be used for mutual communication, but together with mutation, is used primarily for adaptation.

Most operations shown in Table 2 are also self-explanatory. Sequential composition is used when behaviors should be combined in a textual order. Parallel composition is used when behaviors should run in parallel. Cost choice should be used when we are interested in optimization, i.e. it selects the least expensive alternative in accordance with a given cost metrics. Call and definition allow encapsulation behaviors in more complex form, like procedure or function definitions in programming languages. In particular, they allow recursive or iterative repetition of behaviors.

These operations form an operational semantics for composing elementary behaviors of multiple agents into a composite system behavior which can be observed, recognized and controlled by a supervisory agent.

4.1 Control Law Verification Based On Behavioral Equivalence

There are techniques to verify a control specification with respect to requirements specification based on the notion of process equivalence [9]. One attempts to show that a control specification is equivalent to a requirements specification.

Before defining this notion of equivalence, we must first give some notation. We denote the execution of a process P_1, as follows:

$$P_1 \xrightarrow{\alpha_1} P_2 \xrightarrow{\alpha_2} P_3 \xrightarrow{\alpha_3}$$

That is, P_1 first executes atomic step α_1, and evolves to P_2, which executes α_2, evolving to P_3, and so on. Also, let $\mathcal{A}^{\mathcal{E}}$ be the set of all actions or events in the system

Table 1. Process Algebra: Simple Operations

$\alpha ::= (\neg\ \alpha)$		negation
|	$(\$\ P)$	cost
|	$(\rightarrow (\alpha\ \vec{Q}))$	send
|	$(\leftarrow (\alpha\ \vec{X}))$	receive
|	$(\leftrightarrow (\alpha\ \vec{Q}))$	crossover
|	$(\rightsquigarrow (\alpha\ \vec{Q}))$	mutation
|	$(\alpha\ \vec{Q})$	call of (user) defined expression

Table 2. Compositions for Elementary Agent Behaviors

P, Q, P_i ::=	$(\bigcirc_{i\in I} P_i)$		sequential composition
|	$(\parallel_{i\in I} P_i)$		parallel composition
|	$(\biguplus_{i\in I} P_i)$		cost choice
|	$(\sqcup_{i\in I} (\circ\ \alpha_i P_i))$		general choice
|	$(f\ \vec{Q})$		call of (user) defined expression (application)
|	$(:= (f\ \vec{X})\ P)$		recursive definition (abstraction)

including two special behaviors that play a special role in \$-calculus: an empty behavior $\perp$ and an invisible behavior $\mathcal{E}$.

A zero or deadlock $\perp$ represents a predefined process which simply blocks and returns nothing. It is a degenerate process or inaction with blockings. It represents the situation of the behaviors which failed and will never happen, because they are blocked from execution.

A silent (invisible) process $\mathcal{E}$ never blocks, but executes silently returning nothing. Thus it is considered to be unobservable, and therefore uncontrollable by the environment. Although not directly observable, the effects of silent actions can be estimated by cost values. In particular, it can be used to shut down a part of the cost expression, making it invisible or inaccessible for a given agent. It represents a part of the world which is invisible for an agent for some reasons. It may represent a behavior of another agent, not observable by a given agent.

We define *strong bisimilatrity* $\sim$ to be the largest equivalence relation on the class of composite behavior expressions of an agent, such that for any two expressions P and Q, $P \sim Q$ iff, for all $a \in \mathcal{A}^{\mathcal{E}}$, whenever $P \xrightarrow{a} P'$ then for some Q', $Q \xrightarrow{a} Q'$ and $P' \sim Q'$.

Two processes are *strongly congruent* iff they are strongly bisimilar in any context.

Let $\stackrel{t}{\Rightarrow} = (\stackrel{\tau}{\rightarrow})^* \stackrel{a_1}{\rightarrow} (\stackrel{\tau}{\rightarrow})^* \ldots (\stackrel{\tau}{\rightarrow})^* \stackrel{a_n}{\rightarrow} (\stackrel{\tau}{\rightarrow})^*$, when $(x)^*$ denotes the sequential composition of zero or more occurrences of x.

We define *observation congruence* = to be the largest equivalence relation on the class of composite behavioral expressions, such that any two expressions P and Q, $P = Q$ iff, for all $a \in \mathcal{A}^{\varepsilon}$,

whenever $P \stackrel{a}{\Rightarrow} P'$ then for some Q', $Q \stackrel{a}{\Rightarrow} Q'$ and $P' \approx Q'$, i.e. each action of one process is matched by a sequence of actions of another process with the same visible content (are bisimilar $\approx$), and, additionally, each initial action of P or Q is matched by at least one action of the other.

Strong and weak (observation) congruencies equate processes which evolve and deadlock in the same way. They differ in the approach to invisible action ε and requirement for matching of first actions. Different types of equivalences are important because equivalent processes should have the same cost.

5. Controllability and Completeness

This section presents necessary and sufficient conditions for controllability in a multilevel hierarchical network of interacting automata. If T, S and R represent the three levels of autonomy in the network, as shown in Figure 2, then we define 2-level controllability as follows:

Definition 5.1: The language $K \subseteq L(S/R)$ is *2-level controllable* (with respect to S/R) if

1. K is *controllable* (with respect to R) [17], that is $\overline{K}\,\Sigma_u \cap L(R) \subseteq \overline{K}$;
2. $f(K)$ is *controllable* (with respect to S/R), defined as $f(\overline{K})\,\Gamma_u \cap f(L(S/R)) \subseteq f(\overline{K})$; and
3. $\forall s \in \overline{K}$, and $\forall \sigma_i \in \Sigma_{ci}$, for some $1 \le i \le n$, where $s\sigma_i s_i \in (L(S/R))$, (i.e., $\forall s\sigma_i \in \overline{K}\,\Sigma_c \cap L(S/R)$), $s\sigma_i \in \overline{K}$ if and only if $\exists\, s_i \in \Sigma^*_i$ such that $s\sigma_i s_i \in (L(S/R))$, $f(s\sigma_i s_i) = f(s)\gamma_i$ for some $\gamma_i \in \Gamma_i$, and $f(s)\,\gamma_i \in f(\overline{K})$.

To avoid deadlock, it is necessary that all supervisors be complete. For generator R_i, supervisor S_i is said to be *complete* (with respect to R_i) [17] if for all $s \in \Sigma^*_i$ and $\sigma \in \Sigma_i$, whenever $s \in L(S_i/R_i)$, $s\sigma \in L(R_i)$ and $\sigma \in \phi\,(\xi_i\,(x_{oi}, s))$, then it is true that $\xi_i\,(x_{oi}, s\sigma)$, is defined. This condition guarantees that, if an enabled symbol is generated by R_i, then S_i can respond to it and therefore not halt. This notion of completeness extends directly to the top-level supervisor T.

The following theorem [13] proves that *2-level controllability with respect to S/R* is a necessary and sufficient condition on any closed $K \subseteq L(S/R)$ for the existence of a complete top-level supervisor T such that $K=L(T/S/R)$.

Theorem: *Given n generators R_i, $1 \le i \le n$, with disjoint alphabets, each with a complete first-level supervisor $S_i=(S_i, \phi_i)$, let K be a subset of $L(S/R) = \| L(S_i/R_i)$. There exists a complete supervisor T such that $L(T/S/R) = K$ if and only if K is closed and 2-level controllable (with respect to S/R).*

6. Conclusion

This paper outlines a constructive approach to on-line control of dynamic C^2 plans. A formal model is developed based on discrete event behavior control of hierarchical networks of autonomous agents. It supports multiple layers of autonomy which is essential for modeling both deliberative and reactive aspects of combat. Supervisory control is modeled at a higher level of fidelity than the execution level. Planning at the highest levels provides direction as detailed behaviors are concurrently planned and executed at lower levels of the hierarchy.

A useful aspect of this approach is that hierarchies may not be fixed. In addition to the individual participants modifying their plans to adapt to a volatile situation, the structure of the hierarchy is free to adapt as well.

7. Acknowledgments

Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) under agreement number F30602-99-1-0547 and Office of Naval Research under contract number N00014-96-1-5026. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

8. References

[1] Bergstra, J.A. and J.W. Klop, "Algebra Of Communicating Processes With Abstraction," *Journal of Theoretical Computer Science*, 37: 77-121, 1985.

[2] Dean, T. and Boddy, M., "An Analysis of Time-Dependent Planning", in *Proc. Seventh National Conference on Artificial Intelligence*, Minneapolis, MI p 49-54, 1988.

[3] Eberbach, E., R.R. Brooks, "$-Calculus: Flexible Optimization and Adaptation under Bounded Resources in Real-Time Complex Systems," *New Generation Computing,*

OHMSHA, Ltd and Springer-Verlag, Tokyo, Japan, to be published 2000.

[4] Garcia, H. "A Reconfigurable Hybrid Supervisory Control System," *Thesis in Electrical Engineering*, The Pennsylvania State University, 1993.

[5] Garvey, A., Lesser V., "Design-to-Time Real-Time Scheduling," *IEEE Transactions on Systems*, Man, and Cybernetics, 23(6), 1491-1502, 1993.

[6] Hopcroft, J.E. and J.D. Ullman, "Introduction to Automata Theory, Languages and Computability," *Addison-Wesley*, Reading, MA 1979.

[7] Hoare, C.A.R., "Communicating Sequential Processes," *Prentice-Hall*, 1985

[8] Horvitz, E., "Reasoning about Beliefs and Actions under Computational Resources Constraints," in *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence*, Seattle, WA, 1987.

[9] Lee. I., H. Ben-Abdallah, and J. Choi, "A Process Algebraic Method For The Specification And Analysis Of Real-Time Systems," In *Formal Methods For Real-Time Computing*. John Wiley and Sons Ltd., 1996.

[10] Liu, J.W.S., Lin k.J., Shih W.K., Yu, A.C., Chung J.Y., Zhao, W., "Algorithms for Scheduling Imprecise Computations," *IEEE Computer*, 24, 58-68, 1991.

[11] Milner, R., "Communication and Concurrency," *Prentice-Hall*, 1989

[12] Peluso, E., J. Goldstein, S. Phoha, S. Sircar, M. Yukish, J. Licari and I. Mayk. "Hierarchical Supervision For The Command And Control Of Interacting Automata, in the *Proceedings of the Symposium on Command and Control Research*, pages 623-636, Monterey, CA, June 1994.

[13] Peluso, E., "A Hierarchical Structure of Interacting Automata for Modeling Battlefield Dynamics: Controllability and Formal Specification," Ph.D. Dissertation. *Department of Computer Science, The Pennsylvania State University*, 1996.

[14] Phoha, S., Sircar, S., Ray, A., Mayk, I., "Discrete Event Control of Warfare Dynamics," The Technical Proceedings of the *1992 Symposium on Command and control Research and the 9th Annual Decision Aids Conference*, Monterey, CA 8-12 June 1992.

[15] Phoha, S., Sircar, S., Ray, A., Mayk, I., "Computational Grammars for Hierarchical Command and Control Automata, " Tech Proceedings of the *1993 Symposium on Command and Control Research*, Basic Research Group, Jun 1993.

[16] Phoha, S., Peluso, E., Eberbach, E., Kiraly, A., "Coordination of Engineering Design Agents for High Assurance in Complex Dynamic System Design", Invited paper for *Special Track on High Assurance in Intelligent Systems, 3rd IEEE High Assurance Systems Engineering Symposium*, Washington, DC, November 13, 1998.

[17] Ramadge, P.J., Wonham, W.M., "Supervisory Control of a Class of Discrete Event Processes," *SIAM J. Control and Optimization*, vol. 25, No. 1, January 1987.

Stability and Controllability Analysis of Fuzzy Petri Net JFACC Models

R. R. Brooks, S. Phoha, E. Peluso
Information Science and Technology Division
Applied Research Laboratory
The Pennsylvania State University
P. O. Box 30
State College, PA 16804-0030
Tel. (814) 863-1131
Fax. (814) 863-1396
Email: rrb@acm.org

Abstract

Petri Nets are an established method for modeling automated systems [9], [16], [24]. They are concise representations of systems with resource and order constraints. Well-established extensions of Petri Nets model stochastic processes, and dynamic processes with timing constraints [5], [13]. Recently, Fuzzy Petri Nets (FPN's) have been proposed [7], [2], [6], [18]. We use FPN's to model air campaign Command and Control (C^2). This paper discusses stability and controllability of C^2 discrete event systems modeled using FPN's. Controllability is an extension of the definition for C^2 systems modeled using finite automata in [15]. A new definition of stability is proposed, which is consistent with definitions in the literature for discrete event systems modeled using finite state automata.

Keywords: *Discrete Event Systems, Fuzzy Petri Nets, Stability, Controllability, C^2.*

1. Introduction

Military command and control (C^2) differs from traditional control theory. Control theory creates systems that respond to inputs using actuators, where both inputs and actuators are subject to bounded random noise [4], [23]. C^2 deals with warfare, which is characterized by deception and inadequate information [20], [22]. War is frequently won by opponents using new, unexpected strategies [21], [8]. Automated systems are ill suited to evaluating deception and unexpected responses. These reasons, and the need for personal accountability in making life-or-death decisions, constrain practical military C^2 systems to human-in-the-loop approaches.

In the context of the DARPA JFACC initiative, we use Fuzzy Petri Nets (FPN) for human-in-the-loop planning of joint forces air campaigns. This paper briefly describes FPN models for JFACC. It proposes definitions of stability and controllability for FPNs.

2. Fuzzy Petri Net

This section gives a brief introduction to Petri Nets, for thorough introductions see [9], [24]. Formally, a Petri Net is defined by $\{P, T, I, O, M_0\}$ where:

- P is a finite set of places.
- T is a finite set of transitions.
- I is a finite set of directed arcs from places to transitions.
- O is a finite set of directed arcs from transitions to places.
- M_0 is an initial marking.

Figure 1 represents a producer-consumer problem with a channel of only two buffers. One buffer is full in the marking shown. Figure 1 shows the standard graphical representation of the Petri Net. It has four places, three transitions and four markers: $\{P=\{P1, P2, P3, P4\}, T=\{T1, T2, T3\}, I=\{P1T1, P2T2, P3T1, P4T3\}, O=\{T1P2, T2P3, T2P4, T3P1\}, M_0=[2,1,1,0]\}$. Places are circles. Transitions are bars. Directed arcs are arrows. Tokens are black dots. The marking is a vector of integers showing the number of tokens in each place

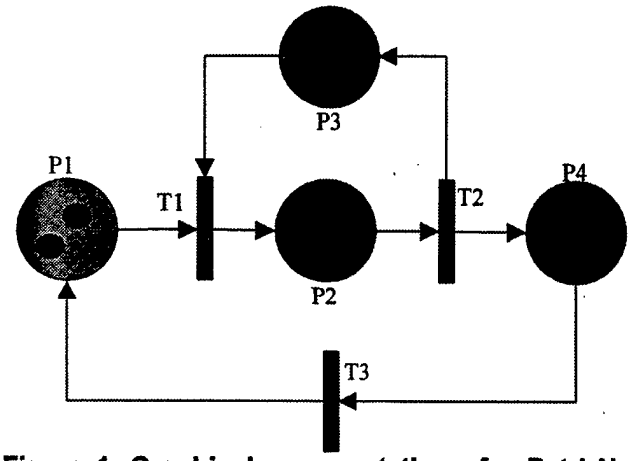


Figure 1. Graphical representation of a Petri Net for a producer-consumer problem. Two buffers exist. One buffer is occupied.

System-state is identified by the current marking. When all places with arcs leading into a transition contain at least one marker, the transition is active. In figure 1, transitions T1 and T2 are active. Since P4 has no marker, T3 is not active. At a given instant, any active transition can fire. Only one transition fires at a time. Any firing order is possible. When a transition fires, one marker is removed from each place with an arc from the place to the transition, and one marker is added to each place with an arc from the transition to the place. For example if transition T2 fires in figure 1, the new marking is M=[2,0,2,1]. Similarly, if transition T1 fires the new marking would be [1,2,0,0].

This simple technique allows modeling of the resource constraints, pre-conditions, and post-conditions of actions. To support real-time system modeling, timing constraints like maximum and minimum time needed can be associated with transition firings [5]. Alternatively, stochastic extensions of Petri Nets can represent Markov models [9]. Both extensions are supported in our model but outside the scope of this paper.

In an adversarial situation, such as an air campaign, not all desirable information is available. In addition, the available information is often uncertain. This is due to measurement noise, jamming, intentional deception, etc. Fuzzy logic is a flexible methodology for expressing and reasoning with uncertain information. We use fuzzy logic to approximate uncertainty, noise, deception, and stochastic processes. See [10],[3] for a full description of fuzzy logic, fuzzy variables, and fuzzy inferencing. Many extensions to Petri Nets including fuzzy variables and fuzzy inferencing have been proposed [7], [2], [6], [18].

In our model, tokens may be either crisp or fuzzy. Crisp tokens are known with certainty to be either present or absent. A fuzzy token is present with a certainty in the range [0..1], where 0 (1) is absent (present) with certainty. How fuzzy values are assigned to tokens for the initial marking depends on the entity being modeled. Where sufficient information is available, the fuzzy membership function can approximate a known statistical distribution. Many factors in a battle environment are not known with absolute certainty, especially information about enemy forces.

Transitions can also be either crisp or fuzzy. They generally represent state changes driven by controllable or uncontrollable events. Since the FPN models possible consequences of a situation, many events may occur with differing probabilities. This is obviously true for uncontrollable events like enemy actions, but is also true for controllable events like friendly actions. The use of fuzzy transitions and fuzzy markers is explained in [2]. It has also been implemented using Higher-Level Petri nets in [17]. In our model, transitions can also be defined recursively by lower level or atomic Fuzzy Petri Nets. Transitions are also associated with additional information such as the entities involved in the transition, controllability, time constraints, etc.

3. Karp-Miller (Coverage) Tree

One advantage of using Petri Nets to model systems is the automatic identification of undesirable properties such as deadlock and livelock. For many methods, the first step in this process is constructing a Karp-Miller tree [9], also referred to as a coverage tree [24]. The Karp-Miller tree is a finite state automata, where one state exists for each possible marking the Petri Net can reach from the original marking. Figure 2 shows the Karp- Miller tree corresponding to the Petri Net in Figure 1. Note that the only two states reachable by taking one transition from the original marking [2,1,1,0] are [2,0,2,1] and [1,2,0,0], as discussed in section 2.

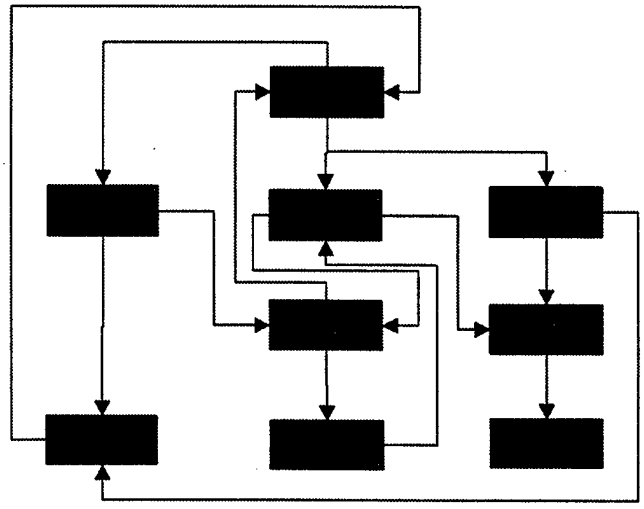


Figure 2. The Karp-Miller (coverage) tree for the Petri Net in figure 1.

An algorithm for constructing Karp-Miller trees is in [24], which also explains how one state can express the unbounded growth of the number of tokens in a place. The

number of states in the tree is bounded, but increase exponentially with the number of places in the Petri Net. The Petri Net is more concise, expressive, and easily constructed than the Karp-Miller tree. The Karp-Miller tree is useful, however, in analyzing the Petri Net for deadlock, livelock, having a bounded number of markers, and the existence of internal invariants [9].

In contrast with traditional Petri Nets, creating the associated Karp-Miller tree of a Fuzzy Petri Net can be problematic. The number of tokens in a place may not be a discrete value. Transitions are no longer Boolean but fuzzy. Our extension of Karp-Miller trees uses α-cuts [3] to maintain a finite number of states in the tree. An α-cut transforms a fuzzy variable by imposing a threshold α. α-cuts are associated with fuzzy transitions to eliminate events that are too improbable. When a fuzzy marker is present in place P*N* (ex. P1), the states in the Karp-Miller tree use a range to indicate the number of markers in P*N* (ex. [0.3..0.6]). Each state in the tree represents a qualitatively different condition of the FPN.

4. Controllability

This section expands on results originally presented in [15], where military C^2 is viewed as a discrete event system. The discrete event system is modeled as a strict hierarchy of communicating finite state machines (FSMs). Lower-level FSM's are supervised by higher-level FSM's with the ability to enable and disable controllable behaviors of the lower-level FSM's. This approach resembles our recursive definition of FPN's where lower-level FPN's can be transitions in higher-level FPN's. Both approaches support multiple levels of abstraction.

In [15], a supervisor recognizes situations in lower-level FSM's and enables or disables controllable behaviors of lower-level FSM's. This restricts the system as a whole, so that a desirable language *K* is generated. *K* always generates strings that end with the FSM in one of a set of (desirable) marked states. The system has both controllable events (actions of friendly forces) and uncontrollable events (actions beyond the control of friendly forces, e.g. actions of enemy forces, equipment failures, environmental conditions, etc.). A supervisor exists if and only if the language *K* is:

(1) Markable - if a string *s'* produces the same output from the FSM as a string *s* in *K* then *s'* is in *K*.
(2) 2-level controllable - The evolution of strings in K as well as their higher-level images are not taken out of the desired language K by controllable or uncontrollable events.

In the FPN model, we construct the Karp-Miller tree from the FPN and consider the Karp-Miller tree as a C^2 FSM. The FPN is *controllable* if and only if a supervisor exists for its Karp-Miller tree. The supervisor is capable of restricting system output to the language *K*.

Intuitively, the system is *controllable* if and only if a strategy exists leading the system within constraints to a desirable end state responding to any possible sequence of uncontrollable actions.

5. Stability

In the literature, [12] and [11] define a stable FPN as one with no unstable states. An unstable state is defined as a marking of the FPN for which no transition exists. This is equivalent to the existence of a state with no transitions in the associated Karp-Miller tree (i.e. a deadlock exists). We do not use this definition, since it is too narrow for our application. For example if the deadlock state is a desired end-state, the system would still be stable for our application.

In discrete event systems, stability has traditionally been associated with the ability of a system to tolerate a limited amount of illegal behavior. In [14] for infinite string inputs, stability is defined as the system visiting a given set of states infinitely often. In [19] a discrete event system is considered as stable if it is guaranteed to reach one of a predefined target states within a finite set of steps.

We propose a definition of stability for our FPN model consistent with discrete event system definitions. An FPN is *stabilizable* if, in response to any finite set of inputs, there always exists a path to one of the desired end states in its Karp-Miller tree.

Stabilizability is verified by performing connectivity analysis on the Karp-Miller tree. An algorithm for this can be found in [1]. Each strongly connected component in the Karp-Miller tree must contain a desirable end state. If this is not the case, a sequence of events exists from which there is no possible return to a desirable end state. This is an *unstable FPN*, and clearly represents an undesirable characteristic.

The existence of a desirable end state in every strongly connected component means in response to any finite string of inputs, it is always possible to return to a desirable end state in a finite number of steps. To phrase this in the same manner as the controllability definition: from any state in the Karp-Miller tree it is possible to create a string whose suffix is the same as the suffix of a string in *K*.

6. Conclusion and Ongoing Research

Petri Nets are a useful tool for concisely expressing systems with resource and timing constraints. Among their advantages is the ability to automatically verify system correctness. Fuzzy logic extensions of Petri Nets are necessary to express several aspects of military missions. In this paper, we have shown how Fuzzy Petri Net models can be used to verify the stability and controllability of discrete event systems.

Further research is needed in many aspects of this approach. Efficient algorithms for verifying controllability and constructing Karp-Miller trees using α-cuts are needed. The concept of observability is also undefined. In the

context of air campaign planning, the observable system is likely to be a subset of the entire system.

7. Acknowledgments

Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-99-1-0547. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government.

8. References

[1] Aho, A. V., J. E. Hopcroft, and J. D. Ullman, "The Design and Analysis of Computer Algorithms". *Addison-Wesley*, Reading, MA, 1974.

[2] Ammar, H.H. and L. Yu, "Fuzzy Marking Petri Nets: Concepts And Definition", *Proceedings Of The 1995 IEEE International Symposium On Intelligent Control*, Page(S): 291 - 297, 1995.

[3] Douchon-Meunier, B. "La Logique Flou," *Presse Universitaires De France*, Paris, 1993.

[4] Brogan, W. L. "Modern Control Theory," *Prentice-Hall*, Englewood Cliffs, NJ, 1991.

[5] Bucci, G., M. Campanai, and P. Nesi, "Tools For Specifying Real-Time Systems," *Real-Time Systems*, 8, Pp. 117-172, 1995.

[6] Cao, T., A. C. Sanderson, "Intelligent Task Planning Using Fuzzy Petri Nets," *World Scientific*, Singapore, 1996.

[7] Chen, S.-M., J.-S. Ke, and J.-F. Chang, "Knowledge Representation Using Fuzzy Petri Nets," *IEEE Transactions On Knowledge And Data Engineering*, Vol. 2, No, 3, Pp. 311-319, Sept. 1990.

[8] Czerwinski, T., "Coping With The Bounds: Speculations On Nonlinearity In Military Affairs,". *National Defense University*, Washington, DC, 1998.

[9] David, R. and H. Alla, "Du Grafcet Aux Reseaux De Petri," *Hermes*, Paris, 1989.

[10] Dubois, D. J. "Fuzzy Sets And Systems," *Academic Press*, Chestnut Hill, MA, 1980.

[11] Furuhashi, T.; Kakami, H.; Peters, J.; Pedrycz, W. "A Stability Analysis Of Fuzzy Control System Using A Generalized Fuzzy Petri Net Model," *Fuzzy Systems Proceedings, 1998. IEEE World Congress On Computational Intelligence.* Volume: 1, pp 95-100, 1998.

[12] Hasegawa, T.; T Furuhashi, and Y Uchikawa, "Stability Analysis Of Fuzzy Control Systems Using Petri Nets," *1996 Biennial Conference Of The North American Fuzzy Information Processing Society*, Page(S): 97 -101, 1996.

[13] Laplante, P. A., "Real-Time Systems Design And Analysis," *IEEE Press*, Los Alamitos, CA, 1997.

[14] Ozveren, C. M., And A. S. Willsky., "Stability And Stabilizability Of Discrete Event Systems," *Journal Of The ACM*, 38(3):730-752, July 1991.

[15] Peluso, E., "A Hierarchical Structure Of Interacting Automata For Modeling Battlefield Dynamics: Controllability And Formal Specification," *Ph.D. Dissertation. Dept. Of Computer Science, The Pennsylvania State University*, 1996.

[16] Reutenauer, C. "Aspects Matheamtiques Des Reseaux De Petr,". *Masson*, Paris, 1989.

[17] Scarpelli, H., F. Gomide, and R. R. Yager, "A Reasoning Algorithm For High-Level Fuzzy Petri Nets," *IEEE Transactions On Fuzzy Systems*, Vol. 4, No. 3, Pp. 282-294, August 1996.

[18] Shen, V. R. L. and F. Lei, "Requirements Specification And Analysis Of Digital Systems Using Fuzzy And Marked Petri Nets," *IEEE Transactions On Systems, Man, And Cybernetics-Part B: Cybernetics*, Vol. 28, No. 5, Pp. 748-754, October 1998.

[19] Takai, S., T. Ushio and S. Kodama, "Stabilization And Blocking In State Feedback Control Of Discrete Event Systems," *Discrete Event Dynamic Systems: Theory And Application*, 5:33-57, 1995.

[20] Tzu, S. "L'Art De La Guerre," *Flamarrion*, Paris, 1972.

[21] Van Creveld, M. L. "Command In War," *Harvard University Press*, Cambridge, MA, 1986.

[22] Von Clausewitz, C., "Vom Kriege," *Ferd. Duemmlers Verlag*, Bonn, Germany, 1972.

[23] Zhou, K., J. C. Doyle, and K. Glover "Robust And Optimal Control," *Prentice-Hall*, Upper Saddle River, NJ, 1996.

[24] Zurawski, R. and M.-C. Zhou, "Petri Nets And Industrial Applications: A Tutorial," *IEEE Transactions On Industrial Electronics*, Vol. 41, No. 6, Pp. 567-583, December 1994.

A Power-Market Model Including Liquidity Constraints

Blaise Morton
Honeywell Technology Center
3660 Technology Drive
Minneapols, MN 55418
morton_blaise@htc.honeywell.com
phone 612-951-7295, fax 612-951-7438

Grant Erdmann
University of Minnesota, Mathematics
Vincent Hall
206 Church Street SE
Minneapolis, MN 55455
erdmann@math.umn.edu

Abstract

In a recent research project funded by EPRI, Honeywell began development of an interactive testbed for modeling electric power markets. The ultimate goal was a flexible tool, designed so that users could create their own models using intelligent agent templates. To help demonstrate the capabilities of the tool we defined a basic set of agents, some of which are described in this note. The focus of this paper is on a key liquidity constraint which we believe should be incorporated into the financial strategies of the various agents to make the simulation behave in a realistic way.

The specific market addressed here is the futures market for electric power. Companies involved in commodity-intensive businesses like oil refining, power generation and finance rely on trading strategies in futures markets to hedge their risks. The profitable survival of these companies depends on the effectiveness of their hedging strategies. Market-model based tools for generating hedging strategies are usually based on a tradeoff between profit and risk, with little attention paid to liquidity. As recent market behavior has shown, better tools that account for liquidity are needed. Here we present an approach in which profit variance (risk) and liquidity play distinctive roles in a constrained stochastic optimization. Some possible consequences of liquidity constraints are illustrated with an example simulation.

1 Introduction

Today's increasingly competitive market for electric power has brought about new trading contracts involving brokers and power exchanges. Though heightened competition may ultimately benefit the consumers of power, the producers of power will certainly find their business environment more challenging in the near term. When power prices spiked in the early summer of 1998, for example, several utilities were forced to report significant losses because of counterparty defaults [1]. Companies faced with such a volatile, competitive market need simulation testbeds on which to test their trading strategies.

As part of a research project funded by EPRI, Honeywell Technology Center and the University of Minnesota began development of SEPIA, a "scenario-free" modeling and optimization tool for the electric power industry [2]. The tool was designed to help the electric power industry gain insight into decision and control strategies in a competitive, deregulated environment. A prototype version of the tool was completed before the first phase of the project ended. Although only a simple market/economy model was incorporated, this early version of the tool provides a framework within which more realistic schemes, such as the one detailed in this paper, could be implemented.

Some key features that were to be demonstrated were:

1. Power producing and consuming agents driven by realistic business goals
2. Stochastic economic models of the type used in mathematical finance
3. Imperfect forecasts used by the corporate agents to plan trading strategies
4. One or more exchanges on which power futures can be sold
5. Transmission agents to reflect realistic constraints and costs

We believe all these features are essential if one is to create a useful, realistic model for power trading. In this note we concentrate on how the first four of them may be implemented. Transmission issues and bilateral agreements will be discussed in another article, here we focus on power trading within a single zone in a single futures exchange where transmission is not an issue.

2 Structure of Agents

The four types of agent in the basic market scenario discussed here are: power producing companies (PPCs), power consuming entities (PCEs), a power exchange and the economy/environment. The structure of the first three of these is discussed in this section, while the structure of the economy/environment is summarized in the next.

Each agent in the simulation has a specific set of functions, external I/O interfaces and internal data files. The strategies of the agents are realized by means of internally stored business plans and externally targeted messages to other agents (business deals). Agents interact with each other by sending and receiving messages, electric power and cash.

Corporate agents such as power producing and consuming companies have business objectives that are reflected in their internal dynamics in the simulation. These business objectives are discussed in sections 4 and 5 below. Non-corporate agents such as the power exchange, residential loads and the economy/environment have less complex internal dynamics.

2.1 Power Producing Companies

Power Producing Companies are represented by corporate agents that produce power for sale on the exchange. Each of these agents has five departments: production, marketing, procurement (purchasing), finance and strategy. There are 4 important market-oriented plans (schedules) and budgets associated with the four specialized departments:

1. procurement has a plan and budget for purchasing fuel
2. production has a plan and budget for generating power
3. marketing has a plan and budget for selling power
4. finance has a plan and budget for accounts payable/receivable and available credit

Also, each specialized department maintains an internal (read-protected) file

1. procurement has a fuel inventory file
2. production has a generator schedule file
3. marketing has a file of contracts with the power exchange
4. finance maintains the file of financial records

The power-producing agents receive, as inputs from the simulation environment, forecasts from the economy/environment agent and quoteboard prices from the power exchange agent. On the basis of these external inputs and internal corporate files, the strategy department evaluates the four market-oriented plans and budgets and provides direction to each department for their revision.

2.2 Power Consuming Entities

There are three types of power consuming entity: industrial, commercial and residential. As a first approximation, the commercial and residential entities are little more than stochastic loads parametrized by the state of the economy/environment agent. The industrial entities, also called power consuming companies, have internal departments and structure similar to the power producing companies.

2.3 The Power Exchange

The power exchange is a special agent comprised of three departments: exchange management, brokerage and finance. The basic functions of these departments are similar to those of a futures exchange as described in [3].

Exchange management maintains a power-price quote board, keeps track of orders placed by the brokers, executes orders when appropriate and then notifies brokers of executed orders. It also serves as a clearinghouse for the positions open at contract expiration, sending delivery notices to the appropriate counterparties.

The brokerage department communicates with traders in the power producing companies and power consuming entities. Brokerage also keeps track of open positions held in the accounts of its clients and notifies clients when orders are executed.

Finance keeps track of account net equity and margin for each of the clients trading on the exchange. It sends account activity reports, payments and billing notices when appropriate. It also has the responsibility for notifying exchange management and brokerage about restricted accounts.

The power price quote board is basically a table indicating the highest bid and lowest offer for standardized quantities of power for delivery during each specified hour at each specified location. Other data such as opening price, daily high and low, open interest and volume are also available to give traders some idea of market state and conditions.

3 Modeling the Economy and Environment

To determine critical budgets and plans, the corporate agents require forecasts of the economy and the environment. Projected changes in simulated variables such as fuel prices and the weather will affect trading strategies. Some parameters in the environment/economy that should be included are:

1. Fuel costs
2. Weather
3. Interest rates
4. Economic growth
5. Consumer demand for manufactured goods
6. Raw-materials costs for manufactured goods
7. Residential demand for electricity
8. Commercial demand for electricity

With the exception of number 6, these parameters are the ones most often mentioned by management of power-producing companies in their explanations of financial performance in 10-Q and 10-K reports filed with the Securities and Exchange Commission (SEC) [8].

The internal dynamics of the power-consuming companies are influenced by parameters 3-6 above. Industrial demand for electricity will be reached as a consequence of these dynamics in the simulation. One model one may use for these eight parameters is a set of coupled stochastic differential equations of the Vasicek type used in mathematical finance [4], i.e.:

$$dX = A * (B - X)\, dt + \Sigma\, dZ(t) \qquad (1)$$

where X is the vector of the eight states, B is the expected mean of X, A is the mean-reversion matrix, t is time, Σ is the volatility matrix, and $Z(t)$ is a Wiener process. To reflect changes in season or economic conditions the parameters A, B and Σ should be allowed to change with time. Equation (1) may be replaced by any other model – our approach depends only on the assumption that future values of the state vector are represented by a stochastic process.

For our model, forecasts are made available to the corporate agents in the form of corrupted estimates of the A, B and Σ parameters sent periodically by the economy/environment agent. Each agent is sent slightly different forecasts to reflect the differences of opinion held by the various forecasting services used by industry.

This model is not the best for all practical applications, but it illustrates some of the general features. More detailed models of the economy and the environment have been developed for use by power industry experts. Those who have better models can use them in place of the simple one described here.

4 Operating Income, Cash Flow and Liquidity

In our approach we use standard financial statements and generally accepted accounting principles to quantify the performance of the corporate agents. Specifically, we consider the summary balance sheet, income statement and cash-flow statements in their usual forms as found in quarterly and annual financial reports filed with the SEC [8].

When projecting performance for the coming quarter (accounting period), the numbers that will appear in the financial statements are represented by ran-

dom variables. We can think of the financial statements themselves as random vectors. To see how this works, we consider next two important entries in these statements: projected operating income and projected cash flow.

4.1 Projected Operating Income

To compute projected operating income, the first step is to compute projected revenues (a random variable). This can be done by looking at the power production schedule, generator model, quote-board data and the exchange account. The next step is to project the cost of fuel needed to produce the scheduled generation (this projected cost is also a random variable). Add the projected fuel cost to projected operation and maintenance cost, depreciation and amortization cost + general taxes to obtain total operating expenses. Then

$$POI = PR - PTOE \tag{2}$$

where POI is projected operating income, PR is projected revenues, and $PTOE$ is projected total operating expenses. The projected income (net) can be computed from the projected operating income by subtracting the projected interest expenses and the projected income taxes.

The projected net income is also called the projected profit or earnings, which forms the basis of stock pricing by fundamental analysis. According to fundamental analysis, the price of a stock should equal today's book value plus the present value of future profits discounted by an appropriate interest-rate curve. Other things being equal, stockholders like to see the value of their shares as high as possible, so corporate management tries to maximize projected net income.

But other things are not always equal, and maximization of projected net income is not the only rule management lives by. A second important consideration is to keep the risk (possibility of lower than expected profit) sufficiently small. Some companies have higher levels of risk than others by the very nature of their business. Shareholders are more tolerant of risk in a gold-mining concern than they are in an electric utility company, for example. Still, at any company, if quarterly earnings should fall below an acceptable level, the shareholders will fire management and bring in someone new. So management cares about limiting profit risk.

This level-of-acceptable-risk constraint could be formulated in a number of ways. Here we take the traditional approach and use the standard deviation of the projected profit as the measure of profit risk. So the optimization criterion becomes a constrained optimization: maximize the expected value of the projected profit subject to the constraint that the standard deviation of the profit is smaller than ϵ_1.

4.2 Projected Cash Flow

The projected cash flow needs to be computed continuously over the accounting period to come. The cash flow up to time T is defined to be the difference between dollars in and dollars out added for that period up to time T. The projection of the cash flow requires the complete budget and sales projections over the period in question. The cash flow over the quarter can be thought of as a stochastic process (i.e. a time-parametrized random variable).

One simple formula, often a good approximation, is [6]:

$$CF = I + DA - DR \tag{3}$$

where CF is cash flow, I is income, DA is depreciation and amortization, and DR is debt retirement.

4.3 Liquidity

Simply put, liquidity is the ability to pay bills on time. To evaluate liquidity we need to consider two factors: working capital and cash flow.

Working capital is sometimes defined as the excess of current assets over current liabilities. Problems arise with this loose definition when receivable accounts are questionable or when inventories are difficult to liquidate, but the basic idea of working capital is the amount of funds in excess of current accounts payable available for operational needs.

Given a company's working capital $W(0)$ today and the projected cash flow into the future, the working capital $W(T)$ at a future time T is the sum of today's working capital and the cash flow between today and time T. The working capital $W(T)$ at a particular time T in the future is a random variable.

The forecasting of working capital is one of the key analytical steps in determining the credit rating of a corporation. Suppose GenCo wants to borrow money from a bank with plans to repay the sum at time T

in the future. Before lending the money, a banker would want to be comfortable about GenCo's ability to survive from now until time T and then repay the loan.

If projected cash flows were certain, the banker might make the loan if working capital $W(t)$ for $0 \leq t \leq T$ satisfied:

1. $W(t) \geq 0$ for $t < T$
2. $W(T) \geq$ loan payoff amount

Given the uncertainties in the real world, however, a careful banker will be much happier if the projected working capital exceeds these bounds by several standard deviations of the projected cash flow at each time $t \leq T$.

Those corporations that have good liquidity (adequate working capital, positive cash flow with low standard deviation relative to working capital) receive high credit ratings and can borrow money at low interest rates. Those companies with poor liquidity have lower credit rating and must pay a higher interest rate for their loans (if they can borrow at all).

So we are led to one practical consequence of liquidity – the cost of borrowing money. Most companies need to borrow money from time to time, and they never have complete control over cash flow, so to insure creditworthiness they might try to keep working capital $W(t) > L$ for some positive constant L at all times t. From a mathematical perspective, because the future is uncertain, this goal is stated as follows: for all time t, keep $W(t) > L$ with probability at least $1 - \epsilon_2$ for some sufficiently small ϵ_2.

4.4 Profit and Liquidity May Be Conflicting Goals

The need for liquidity can have a negative impact on profitability in several ways.

First, no matter how great the profit potential of growth might appear, a company must limit its investment in plant and materials because it has to maintain cash reserves adequate to pay its bills in an uncertain future environment. Many otherwise successful, high-growth companies on the path of rapid expansion have failed because a small perturbation to their operations caused them to run out of money. The liquidity constraint keeps a lid on profits that might be gained through more rapid growth. This relation between cash flow and earnings is fundamental. A corollary is the tenet that cash flow growth is a prerequisite for sustainable earnings growth.

Second, poor liquidity means a bad credit rating and higher costs associated with borrowing money. These higher interest-rate costs lower projected income (see discussion, section 4.1).

Third, there are times when a company must operate at inefficient levels of production (negative marginal profit) in order to generate cash flow. This situation can be so severe that the company might have to take a loss (negative net income) for a period in order to bring in enough money to pay bills. Though it might seem paradoxical that one can bring in more money by losing more money, the rules of accounting make this possible. By equation (3) we see that if no debt is retired or created, cash flow exceeds income by the amount of depreciation and amortization. Thinking of these variable as functions of production output, if depreciation and amortization increase at a greater rate than income decreases, the marginal cash flow may be positive while marginal income is negative. A power-generating company may find itself in exactly this situation if it has a big payment (e.g. bond redemption) coming due and, to raise the cash, it has to operate its least profitable generators at a time when electricity prices are low.

5 Baseline Optimization Problem

The corporate agents have the following functions: purchasing, production, marketing, finance and strategy. All of these functions are integrated in a planning/budgeting process, which needs to be formalized before implementation in the simulation. The baseline concept for corporate strategy is to maximize expected profit subject to two constraints:

1. the standard deviation of the profit is smaller than ϵ_1
2. the probability of working capital falling below L is smaller than ϵ_2

In the second constraint, L is the minimum acceptable level of working capital at any time during the quarter – by picking $L = 0$ the condition becomes a solvency constraint.

By trying to solve this constrained optimization problem the corporate agents should generate sensible trading strategies. Note that one might want to add other constraints or objectives (e.g. environmental responsibility) but we believe the problem as stated should give rise to sensible plans for the financial departments of corporate agents.

5.1 Solving the Baseline Optimization Problem

We do not have an elegant, closed-form solution to the optimization problem. Our proposed approach was to use learning strategies in our agents, though a more mundane solution might be possible. The real problem is extremely difficult in practice because forecasts of future conditions are required in the solution process. One can get a rough idea of future conditions by looking at price quotes at the futures exchange, but futures prices are volatile and should not be the only source of future prices used for planning.

To see what is involved in solving the problem, let us look at a highly inefficient and suboptimal but direct approach – basically a random search. Recall that the answer should be in the form of plans and budgets for operations over the next quarter (or other accounting period). Assume given ϵ_1, ϵ_2 and L.

1. Start with an arbitrary (but definite) schedule for producing power and hedging
2. From this schedule, generate the associated plan and budgets for operations (fuel and labor). Assume finance activities are minimal.
3. Use the futures-exchange bid prices (over the quarter) to value the price at which the power can be sold.
4. Project cash flows from operations (sales - costs)
5. Determine whether the projected cash flow satisfies the liquidity constraint over the next quarter (accounting period).
6. If the liquidity constraint is satisfied, compute the projected entries for the profit and loss statement for the quarter
7. If the profit variance satisfies the first constraint, compute the expected profit.

If the conditions in steps 6 or 7 are not satisfied, throw away all the data computed and pick a new power production schedule or a new hedging strategy in step 1. Keep trying different power production schedules and hedging strategies until one is found that works – save all the plans and budgets for that scenario. Call that solution the current candidate. Then keep exploring the set of power-production schedules and hedging strategies, and for those that satisfy the constraints, compare expected profits with the current candidate. If a new solution has better expected profit use it to replace the current candidate. Continue the search until search resources (e.g. computation time) are expended.

5.2 Liquidity and Low Risk May Be Conflicting Goals

Commodity producing companies often use the strategy of selling-short in the futures market a percentage of future production to reduce profit risk. However, by shorting futures contracts to reduce risk, the company simultaneously increases the probability that working capital will drop below the level L. The problem is the daily marking to market of futures positions.

Every day, at the market close, the values of all futures positions are computed, and the net equity of each account is examined. If the equity of an account drops below the maintenance margin level, the party holding that account has to deposit more money in the margin account to keep the positions open. This is a margin call. Because of the obligation to meet margin calls, the hedged positions could drain working capital.

For example, suppose GenCo choses to sell short 20% of its production for the next month in a 1-month futures contract. Suppose it has working capital adequate to pay the current futures market price of 40% of a month's production. If the futures price should quadruple the next day, GenCo would find it has exhausted all of its working capital, and then some, on its short futures position. Because GenCo could not meet the margin call, the exchange broker would close out the positions, taking all the margin-account money and (perhaps) take a note for the rest (or some more extreme measure). Then, if electricity prices drop back down to where they were before, GenCo will have been whipsawed. The attempt to hedge would have cost them 60% of the month's production, and their ability to pay bills would be

severely impaired.

So is one better off hedging production in the forward markets, where daily margin payments are not needed? There are problems of a different nature associated with forwards, however, including inability to reverse a position and the risk of counterparty default [1],[9], so it is not clear how best to hedge. In practice, commodity-producing companies do hedge some of their risk in the futures markets, so we expect the electricity producers will do so as well.

The whipsaw scenario could be avoided by buying puts on the futures contracts instead of selling short, but options must be paid for in cash, and the cost is a drain on working capital. The bottom line is that reduction of profit risk often comes at the cost of increased liquidity risk and vice versa. The practical challenge is deciding how to make the trade.

5.3 Properties of the Optimal Solution

Let us examine some expected properties of solutions to the baseline optimization problem.

First, if forecasts predict that electric power futures prices should increase (a bull market), the maximum-profit objective would oppose the selling of futures as a hedge. In fact, if only the expected profit were considered then futures should be purchased. Here we encounter the profit-risk constraint. The level of risk involved in buying futures based on forecasts is far greater than the risks inherent in operating a stable power-producing business, so a net-long position in futures for a producer is likely to exceed the profit-risk constraint. We might expect aggressive producers to operate hedge-free, but more conservative producers should stay at least partially hedged. The point is, in a bull market, some hedging may be dictated by the profit-risk constraint.

Conversely, in a bear market, the maximum-profit objective would dictate hedging essentially all future production at the current futures prices. Note that this strategy is not unheard-of in business – Barrick Gold Corp. has a history of hedging much of its gold production for years into the future [9]. Profit risk is minimized when all production is hedged [1] , so the profit variance constraint will not prevent a 100% hedge. But now the liquidity constraint might prevent total hedging, for reasons discussed in the previous subsection. In a bear market the liquidity constraint might preclude a 100% hedge (at least with futures), though it might support a partial hedge in the futures market.

There is also the question of whether to hedge with futures contracts or options. In practice, if today's prices allow a comfortable level of profit and market volatility is low (i.e. puts are cheap), companies sometimes hedge their production by buying out-of-the-money put options, giving downside production without limiting gains should prices move higher. If volatility is high they might choose to sell call options (limiting upside potential) to raise money needed to pay for expensive puts. We expect that criteria for deciding whether and how to use options and/or futures contracts to hedge should follow from a solution to the baseline optimization problem.

Finally, let us consider a simple example to see how the optimal strategy might evolve in time.

Assume GenCo is hedged to the maximum level consistent with its liquidity constraint, and profit risk is not a problem. Suppose forecasts show that electricity prices and price variances are likely to move down (just after a temporary spike in energy prices, for example). As time evolves two things happen if no new hedging is employed:

1. As contracts mature, the amount of hedged production decreases
2. The duration (average time to maturity) of the hedged portfolio decreases

The marginal profit with respect to hedging is positive, and it may also be true that liquidity risk is diminishing. If so, the constrained optimal solution should involve incremental replacement of hedged contracts as they mature. An interesting question is how to select the size and maturity of new hedges to replace the expiring contracts as they go off the board.

6 Example Scenario

We now illustrate some of these ideas in a particular scenario. This problem concerns a power plant

[1] risk due to uncertainty in ability to produce must still be accounted for, but this uncertainty is usually lower than the uncertainty in market prices

which wishes to hedge its position by selling futures contracts on its power. The goal is to simulate selling power in the futures market to explore the relationships between profit, profit variance, and margin requirements. Here we explain a simple model and give numerical results from Monte Carlo simulations. The idea is to show approximate results of a type we would expect to see in a more detailed SEPIA simulation.

6.1 Assumptions

The price of power is assumed to satisfy

$$dP = \mu P dt + \sigma P dz$$

Hence, at time t, the price of power at time T in the future has the lognormal distribution

$$\log \frac{P_T}{P_t} \sim \phi \left[(\mu - \frac{\sigma^2}{2})(T-t), \sigma\sqrt{T-t} \right].$$

The system parameters were chosen roughly to model the Wyodak plant operated by Black Hills Corporation. The coal mine supplying the power plant is operated by the same company, and the mine is located on the site. Using data obtained from SEC filings we derived the following model parameters:

- The Wyodak Plant is a 330 MW plant [10]. Since a futures contract is based on one-month's production at a rate of 2 MW [12], the Wyodak Plant can produce enough power for 165 contracts.
- The working capital available to Black Hills Corporation is \$13.080 million [11]. Wyodak Resources operates the coal mine located at the plant. Since Wyodak Resources accounts for 31080/313662=9.9% of Black Hills Corporation's operating revenue [11], We assume that the working capital available to the Wyodak plant is 9.9% of \$13.080 million, \$1.30 million.
- The interest rate the plant will pay on short-term loans can be estimated by the rate that it pays on long-term bonds. For the Black Hills Corporation, the highest rate paid on bonds is 9.49% [10]. We assume that this is the rate the plant might have to pay in a revolving credit agreement, although the true rate may be considerably lower.
- We ignore the cost of trades. Large companies have transaction costs of as little as \$10 per contract, which will be considered to be negligible in the simulations.
- Tax considerations will be ignored.

Using data for electric power futures contract, we can make several reasonable assumptions:

- We will consider only the California-Oregon border power contracts. The power will not necessarily be delivered at the California-Oregon border. However, we allow for a constant geographic basis to be added, which would alter the revenue by a constant.
- The futures contract price is \$32.75 per Mwh (as it was for December 1998 contracts on November 17, 1998), which gives \$24104 per 736 Mwh contract.
- The initial margin is 3% of the purchase price, as is the maintenance margin.
- Futures contracts will be sold for the first 12 months of operation, with delivery being made for the month after contract expiration. Each month is taken to be 21 business days (consistent with the terms of the NYMEX futures contract).
- Using historical data from the New York Mercantile Exchange for futures contracts on California/Oregon Border power, we estimated the σ for the price distribution. Data from September 1, 1998 until November 16, 1998 (on the December 1998 futures contracts), as well as data from November 2, 1998 until January 18, 1999 (on the March 1999 futures contracts) were used to estimate σ. The values for σ were found to be 0.3366 and 0.4163, respectively. We assume that a good estimate of σ is the average, 0.3765.
- $\mu = 0$ in the price distribution. This means that, if adjusted for inflation, the price of power will, on average, stay the same.
- The plant can borrow money when the working capital is exhausted. The continuously compounded interest rate charged is 9.49%. These loans will be repayed as money becomes available.

6.2 Computational Results

Monte Carlo simulations of this model were run, with 50000 trials per number of contracts. All monetary values are normalized by the average value of one year's production, $\$24104 \times 165 \times 12 = \$47.7$ million.

Figure 1 shows the expected change in profit for different hedging strategies. This represents interest rate charges only, as all other losses will be averaged out by the many trials conducted. We see, as expected, that more loans must be taken out if more contracts are sold. We see that the maximum average loss caused by hedging is less than 0.06% of one year's production value, about \$28000. This cost is relatively small.

Figure 2 shows the standard deviation of the profit. We see that this measure of profit risk varies almost linearly with the number of unhedged contracts. The standard deviation of the profit is almost one fourth of a year's production value if no power is sold on the futures market.

Figure 3 shows the average maximum margin required for different hedging strategies. We see that this measure of required liquidity varies almost linearly with the number of hedged contracts. The average maximum margin is about 12% of a year's production value if all power is sold on the futures market.

Figure 4 is a graph of the average maximum margin vs. the standard deviation of profit, which shows the trade–off between liquidity and profit risk. One can use such a graph to determine, given the maximum tolerance for either value, the minimum value of the other quantity. Here, for instance, we see that, if the effects are balanced by setting the values equal to each other, they are about 7.5% of the value of one year's production.

Figures 5–6 show scenario analysis for different cases. The price of power is allowed to vary at certain numbers of standard deviations from average. The change in profit caused by hedging is shown in Figure 5. We see that, by hedging all production in a +3 standard deviation year, one misses a possible 80% gain in total production revenue. However, by hedging all production in a −3 standard deviation year, one can avoid an 80% loss in total production revenue. Figure 6 shows how the maximum margin requirements can increase if the price of power continually rises.

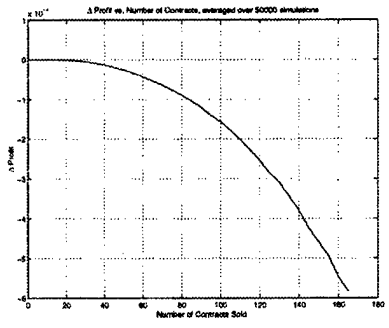


Figure 1: Change in profit for different numbers of contracts, 50000 trials per number of contracts.

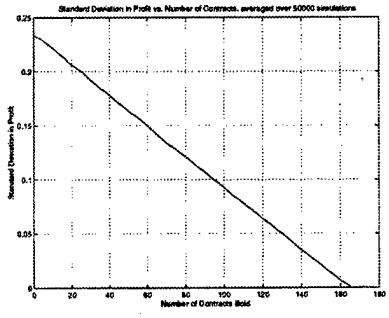


Figure 2: Standard deviation in revenue for different numbers of contracts, 50000 trials per number of contracts.

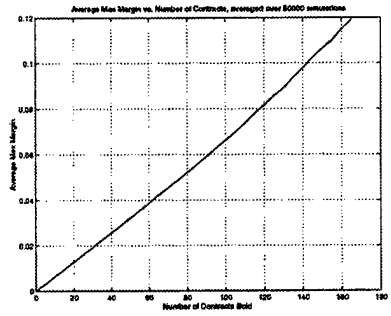


Figure 3: The average maximum margin requirement for different numbers of contracts, 50000 trials per number of contracts.

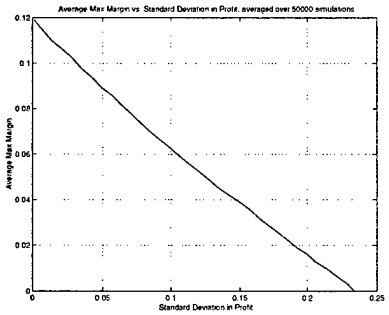


Figure 4: Average maximum margin requirement vs. standard deviation in profit, 50000 trials per number of contracts.

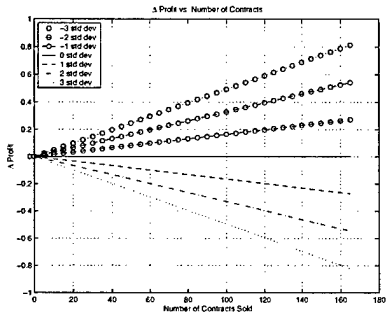


Figure 5: Change in profit, if price of contract varies from average by -3, -2, -1, 0, 1, 2, or 3 standard deviations.

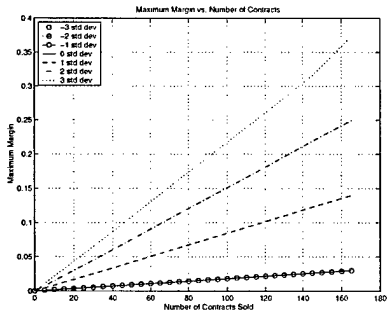


Figure 6: Maximum margin requirement, if price of contract varies from average by -3, -2, -1, 0, 1, 2, or 3 standard deviations.

6.3 Conclusions based on the Example

The given model appears to be reasonable and useful for the real problem.

For the Wyodak plant, the model gives the result that one can reduce the standard deviation of the revenue with a relatively small decrease in the expected profit.

Since about 21.6 million shares of stock for Black Hills Corporation exist [11], hedging full production costs only 0.13 cents per share in an average year.

The model is a good tool for scenario analysis and decision making. If accurate parameters are given, the numerical results and corresponding graphs can be used to determine the behavior of the problem. For a desired average standard deviation in profit or average maximum margin requirement, the approximate cost of hedging and number of contracts to sell can be read from the graphs.

This model should be considered a starting point which gives a company a sense of the trade-off between expected revenue and uncertainty in revenue, as well as the trade between profit-risk reduction and liquidity. It would be a good idea to examine more general hedging strategies with dynamic features not considered here. In any case, additional tests should be made before using the results of this model to a decide the best hedging strategy. The company should simulate the strategy suggested by the model on the most extreme observed behavior of the market. This will give worst-case and best-case scenarios consistent with expectations. Also, the company should consider the basis risk, a result of the changing difference between the futures price and the local price of energy. Testing these two items can be done by examining the price history of the futures market and the internal records of the company.

7 Summary

In this paper we described some of the functional features of an agent-based simulation tool, and ways that such a tool could be used to model the electric power market. Specific examples of three agents designed for such an implementation were discussed.

The emphasis here is on hedging with futures markets, though much of what has been said applies to

hedging in the forward markets as well. The basic idea is to implement standard business-finance models in an event-driven, agent-based tool to simulate realistic market behavior.

One important feature, to enhance the fidelity of the simulation, is a dynamic liquidity constraint in the profit optimization scheme. We explained how the goals and constraints may interact in practical situations and explained how the criteria may predict active hedging.

Results of a Monte-Carlo simulation were presented to illustrate the trade between liquidity and profit-risk reduction using a simple hedging strategy. A more thorough study of market strategies would be possible if the agent-based simulation we designed for this application were developed.

References

[1] K. Kranhold, "Some Electric Utilities Expect to Report That Trading Losses Hurt Earnings," Wall Street Journal, July 6 1998.

[2] L. Pires et al., "Complex Adaptive Strategies: Tools for the Electric Power Industry," Technical Report TR-112816, EPRI, 3412 Hillview Avenue, Palo Alto, CA 94304-1395, USA (in preparation).

[3] S. Kroll and M. Paulenoff, **The Business One Irwin Guide to the Futures Market,** Business One Irwin, 1993.

[4] J. Hull, **Options, Futures and Other Derivatives,** 3rd Edition, Prentice Hall, 1997.

[5] G. White, A. Sondi and D. Fried, **The Analysis and Use of Financial Statements,** John Wiley, 1994.

[6] F. Plewa and G. Friedlob, **Understanding Cash Flow,** John Wiley, 1995.

[7] K. Parkinson and J. Kallberg, **Corporate Liquidity,** Irwin, 1993.

[8] Forms 10-Q were reviewed for the quarter ended 3/31/97 of the following utility companies: Cinergy Corp., Duke Power, NSP, PGE, Texas Utilities

[9] S. Pulliam and R. Smith, "Firms Find Thorny Hedges as Gold Price Soars," Wall Street Journal, October 7 1999.

[10] Black Hills Corporation. Form 10–K405, filed March 11, 1998 with the SEC.

[11] Black Hills Corporation. Form 10–Q, filed August 14, 1998 with the SEC.

[12] New York Mercantile Exchange. "Palo Verde and California/Oregon Border Electricity Futures and Options Contract Specifications." Located at the Web address www.nymex.com/contract/electric/spec.html.

Section 2

Decision Support Systems

IMMACCS: A Military Decision-Support System

Jens G. Pohl
Kym Jason Pohl
Anthony A. Wood
Arthur J. Chapman
CAD Research Center, California Polytechnic State University, San Luis Obispo, California
jpohl@calpoly.edu; awood@cdmtech.com; kpohl@cadrc.calpoly.edu; achapman@calpoly.edu

Abstract

The Integrated Marine Multi-Agent Command and Control System (IMMACCS) is a multi-agent, distributed system, designed to provide a 'common tactical picture' with integrated and meaningful decision-support facilities to authorized operators at any access node. IMMACCS has been implemented as a three-tier architecture that distinguishes between information, logic and presentation. It utilizes an object-serving communication facility with subscription and multi-casting capabilities that is based on the Common Object Request Broker Architecture (CORBA). With an emphasis on application, IMMACCS was designed and implemented in concert with its military users as an integral component of experiments conceived by the Marine Corps Warfighting Laboratory to test emerging concepts in military command and control. It was field tested as the command and control system of record during the Urban Warrior Advanced Warfighting Experiment conducted by the Marine Corps Warfighting Laboratory in Monterey and Oakland, California, March 12 to 18, 1999.

Principal design notions and system components

IMMACCS is an adaptive, distributed, open architecture system that is intended to assist military commanders under battle-like conditions when dynamic information changes, complex relationships, and time pressures tend to stress the cognitive capabilities of decision makers and their staff. IMMACCS incorporates four design notions that are fundamental to its decision-assistance capabilities.

Notion 1: Whereas legacy systems typically process data, IMMACCS processes information. The key to the assistance capabilities of IMMACCS is that the system has some 'understanding' of the information that it is processing. In IMMACCS every entity in the screen display of the battlefield (e.g., road, building, truck, tank, enemy unit, civilian group, etc.) as well as intangible entities such as weather, attack, defense, and so on, are represented as individual objects with characteristics and relationships to each other. Therefore, the military commander and staff officer interacts with a computer display that consists of hundreds of real world entities (objects) that all have some 'understanding' of each other's nature, interests and objectives, and a great deal of 'understanding' of their own characteristics and capabilities.

Notion 2: IMMACCS is a collection of powerful collaborative tools, not a library of predefined solutions. This approach is intended to overcome the deficiencies of legacy systems in which built-in solutions to predetermined problems often differ significantly from the complex operational situations encountered in the real world. IMMACCS is a collaborative decision-support system in which the operators interact with computer-based agents (i.e., decision tools) to solve problems that cannot be precisely nor easily predetermined.

Notion 3: IMMACCS incorporates agents that are able to reason about the characteristics and the relationships of the many real world entities (i.e., objects) that are recognized within its representational schema (i.e., ontology). During its first field test (held in California, March 1999 [1]) IMMACCS included agents capable of providing assistance in decision areas involving: weapon selection and deconfliction; Rules of Engagement; potential fratricide situations; enemy engagements and decision points; and, logistical re-supply requirements. In addition, IMMACCS supports mentor agents that can be dynamically created to represent the interests of warfighters and warfighting machines. Mentor agents are intended to extend the capabilities of Marines at all levels by warning friendly units of hostile intrusions into their territory.

Notion 4: IMMACCS integrates planning, execution and training within one common command and control user environment. The computer-based agents and the IMMACCS users continuously collaborate as they interact with each other in rapidly changing battlefield

situations. In this respect IMMACCS reflects the complexity of the real world where problem solutions must be continuously reviewed as conditions change, and it becomes increasingly difficult and inconvenient to separate planning, re-planning, execution, and training functions into artificially discrete activities supported by different applications.

IMMACCS is one integrated system and not a confederation of loosely linked sub-systems. Its architecture is based on the Integrated Collaborative Decision Model (ICDM) [2] [3] multi-agent system development framework applied by the CAD Research Center previously in engineering design [4] [5], transportation planning [6] [7] and military command and control [8] [9]. In its field testing state in March 1999, IMMACCS consisted of the following integrated components (Fig.1):

A Shared Net communication facility (designed and developed by the Jet Propulsion Laboratory (JPL)) that manages the object-based interactions among the various components on a subscription basis. All IMMACCS components are clients of the Shared Net and indicate their information interests by registering a subscription profile. Whenever, information that is within the subscription of one or more cients (whether military commander or squad leader) becomes available the Shared Net automatically pushes this information into a cache memory area and sends an alert message to the client. In addition, clients may also query for information for which they have not subscribed. Even individual agent sessions are clients to the Shared Net and can therefore take advantage of these efficient communication capabilities.

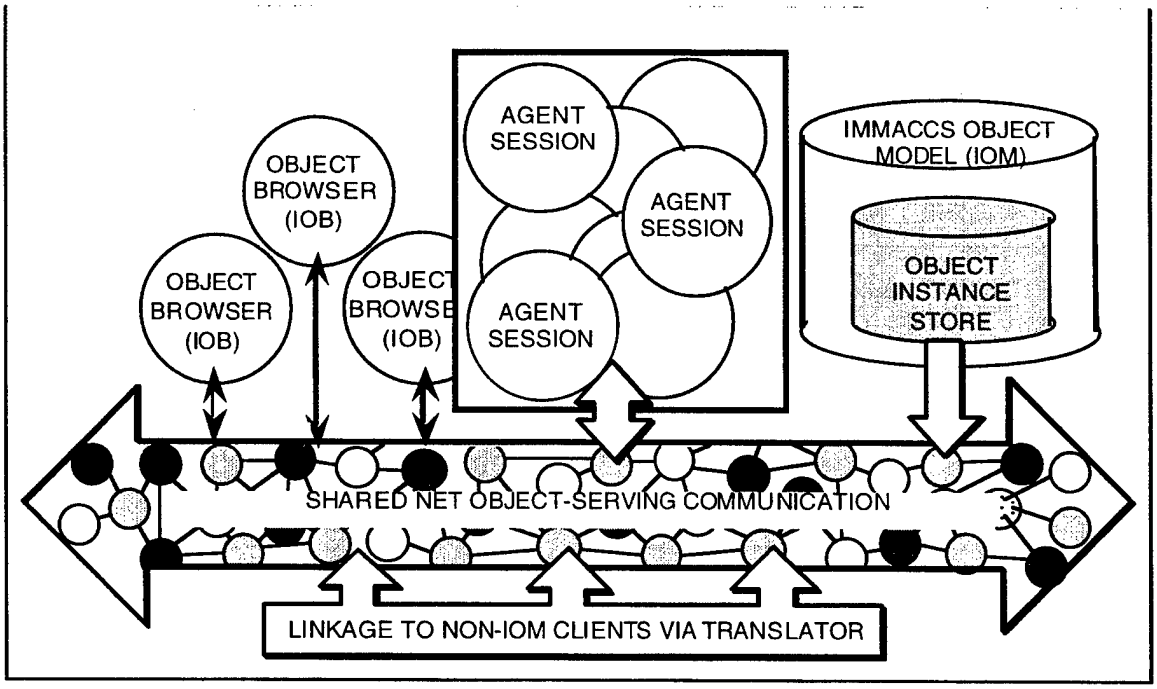


Fig. 1: Schematic representation of the IMMACCS components

An Object Model (designed and developed by the CAD Research Center) that facilitates the internal representation of information (rather than data). In particular, IMMACCS supports the dynamic formation of associations among objects at both the user and agent levels.

An Agent Engine (designed and developed by the CAD Research Center) that automatically initiates an agent session in support of any desired View of the battlespace.

A hardware independent Object Browser (designed and developed by the CAD Research Center) that facilitates user interaction within the object-based information context and the collaborative agent assistance capabilities of IMMACCS. Through the Object Browser the user may: set alert conditions (e.g., request warning of enemy advances to within a user-specified radius of the current position of the operator); obtain agent reports and suggestions; request agent

explanations; explore the location and capabilities of key resources (e.g., local police and fire stations, hospitals, and government buildings) on the object-based infrastructure display of the battlefield; and, enter information to automatically activate any other client(s) of the Shared Net.

A set of Translators (designed and developed by the SPAWAR Systems Center) that are capable of mapping data received from external applications, such as the Joint Maritime Command Information System (JMCIS), to the object-based representation held within the IMMACCS Object Model.

A hardware independent, lightweight 2-D Viewer user interface (designed and developed by SRI International) that connects the Marine in the battlespace via wireless communication to IMMACCS. Each 2-D Viewer hardware device is provided with a differential GPS (Global Positioning System) facility that transmits automatic position reports to IMMACCS. In this way IMMACCS is able to automatically track the current position of all beacon equipped friendly units, and make this information available to agents as they spontaneously and opportunistically reason about events which might affect these units.

A Geographic Infrastructure Database (designed and developed by the Naval Research Laboratory at Stennis Space Center) that provides objectified battlespace infrastructure from NIMA Vector Product Format (VPF) data.

The fundamental requirement of 'information' representation

Although technological advances in computer hardware and communication systems have been truly astounding over the past 20 years, the direct utilization of these advances in the area of decision-support has been less than remarkable. The fact is that we are still using computers largely as ***data*** processing devices that perform only the most menial and least intelligent data transmission and manipulation tasks. While computers are performing these tasks with great speed and accuracy, and while they are able to provide connectivity among a virtually unlimited number of access points, the higher level and much more rewarding tasks of analyzing, interpreting and abstracting data as 'information' and inferring 'knowledge' is almost entirely left to the human users.

This serious deficiency has become increasingly apparent as technological advances have increased computing power, data storage capacities, and data transmission speeds by orders of magnitude in such a short period of time. Convenient global access to users and data has increased the need for information filtering, so that individuals might take advantage of the opportunities for material and personal profit that this connectivity and processing power present to the user. Needless to say, the capabilities of a computer to assist in the intelligent assessment of information are basically non-existent if the computer processes this information as bitmaps and alphanumeric text strings. Any significantly useful human-computer collaborative partnership carries with it the expectation that information is held within the system environment in a representational form that is, if not equivalent to, at least compatible with human cognition.

The current approach for achieving this objective is to represent information in the computer as objects with behavioral characteristics and relationships to other objects [10]. While this approach is hardly sophisticated it does allow real world objects (e.g., airfield, tunnel, building, weapon, tank) to be represented symbolically so that computer software modules can reason about them.

It is important to note that the relationships among these objects are often far more important than the characteristics that describe the individual behavior of each object. For example, the word 'house' holds little meaning if we strip away the many associations that this word represents in our mind. However, such associations to our knowledge of construction materials, our experiences in having lived in houses, and our understanding of how our own home is impacted by external factors (such as rain, sunshine, neighbors, mortgage interest rates, and so on) constitute the rich meaning of the object 'house' [11]. Accordingly, any useful representation of information in the computer must be capable of capturing the relationships among the entities (i.e., objects) in the problem system.

While some of these associations are fairly static (e.g., a weapon is a kind of asset and a lethal weapon is a kind of weapon) many of the associations are governed by current conditions and are therefore highly dynamic. For example, as a platoon of soldiers moves through the battlefield it continuously establishes new associations (e.g., to windows in buildings from which snipers could fire on individual members of the platoon), changes existing associations (e.g., higher levels of risk as the platoon nears an active combat zone), and severs previous associations (e.g., as the platoon is forced to abandon its compromised command post).

Abstract concepts such as privacy, security and power, are less amenable to this approach since their meaning and role in our day-to-day activities is less easily defined. For example, the characteristics of 'privacy' are neither static nor can they be accurately described in relational terms. They depend on a wide range of factors that relate to both environmental and personal circumstances and dispositions. These factors can be only partially accounted for through embedded knowledge and rules, and therefore become largely the

purview of the human members of the collaborative human-computer partnership.

Nevertheless, even with these shortcomings this form of representation of real world objects can provide the basis of usable problem solving support and decision making assistance. Improvements are possible with the addition of knowledge bases and user interaction. In the latter case the user becomes as much a helper to the system as the system serves as an assistant to the user. However, this occurs in quite different ways. The system uses its computing and logical reasoning capabilities to monitor, analyze and evaluate the actions, requests and interests of the user in an opportunistic manner. The user, on the other hand, helps the system to understand the nature of the objects and relationships that it is processing in a more deliberate manner [12].

The need for a high level representation is fundamental to all computer-based decision-support systems. It is an essential prerequisite for embedding artificial intelligence in such systems, and forms the basis of any meaningful communication between user and computer. Without a high level representation facility the abilities of the computer to assist the human decision maker are confined to the performance of menial tasks, such as the automatic retrieval and storage of data or the computation of mathematically defined quantities. While even those tasks may be highly productive they cannot support a partnership in which human users and computer-based systems collaborate in a meaningful and intelligent manner in the solution of complex problems.

In IMMACCS a comprehensive object model representing the characteristics and relationships of the entities expected in the urban battlespace environment, forms the basis of all agent capabilities [9]. Utilizing a partially automated process and a standard graphical methodology (i.e., Unified Modeling Language) it became possible to directly produce final application code. While past efforts by others [13] [14] [15] contributed valuable information relating to the identification and description of object classes for battlefield simulations, these references place little importance on the associations between objects. Accordingly, the development of the IMMACCS object model was required to focus heavily on defining and representing the relationships among battlefield entities.

The IMMACCS system architecture

Based on the ICDM framework (developed previously by tthe CAD Research Center [2] [3]) the IMMACCS model is based on a three-tier architecture that makes clear distinctions between information, logic, and presentation. These tiers are represented by the three major IMMACCS system components; namely: the Shared Net (information tier), the Agent Engine (logic tier), and the IMMACCS Object Browser (IOB) and 2-D Viewer (presentation tier) (Fig.2). Included in the information tier are two additional components. The first of these is the Marine Corps System IMMACCS Translator (MCSIT) providing bi-directional information translation between IMMACCS and external systems (e.g., JMCIS, LAWS, TSCM, etc.). The second system is the Geographic Infrastructure Database (GIDB) responsible for providing geographic infrastructure information (e.g., buildings, roads, etc.) to the other IMMACCS components. Each of these tiers functions in an integrated fashion to form a comprehensive agent-based decision-support execution framework. This framework allows multiple human decision makers to solve complex problems in a collaborative fashion obtaining decision-support assistance from a collection of heterogeneous on-line agents.

The Shared Net information server

Conceptually, the Shared Net (SHN) represents a library of objectified information which clients utilize to both obtain and contribute knowledge. The only difference is that clients can obtain this information in, not only a pull fashion, but can also have the SHN push them information on a subscription basis. Physically, the SHN exists as an enhanced distributed object server based on the Common Object Request Broker Architecture (CORBA) specification.

As the basis of the SHN, distributed object servers are designed to service client requests for information. The knowledge of exactly where the information resides and how it can be retrieved is completely encapsulated inside the object server. This means that clients need not be concerned with which client has what information and in what form that information exists. This feature becomes instrumental in providing an environment where collaborative application components inter-operate in a de-coupled fashion.

Regardless of the native representation of the information, distributed object servers can be used to present information to clients in the form of objects. However, this does not discount the need for information to be modeled as high-level objects in their native form portraying behavior and conveying relationships. While on the surface this representational morphing capability of object servers seems promising, in practice this feature proves to be quite misleading. If the information is not represented at a high level upon its conception, such objectification amounts to little more than wrapping data in communicable object shells. These shells fail to convey any additional insight into the meaning or implication of the information than was present to begin with in its original form. Although in the future there may be potential for successful research efforts in this area, at present, unless information is originally modeled as objects, knowledge-oriented applications prove to gain little from the distributed object server feature.

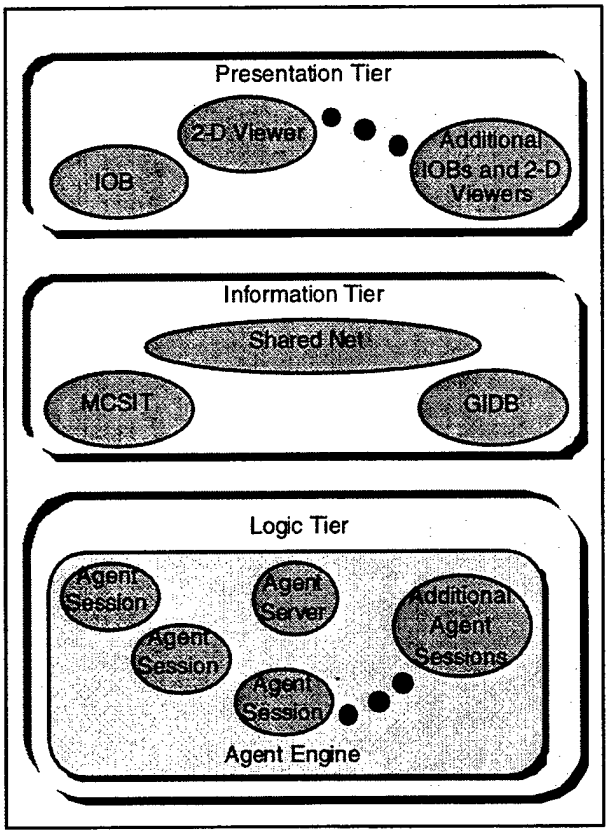


Figure 2. Three-tier architecture of IMMACCS

However, applications such as IMMACCS that do model information as high-level objects stand to gain considerably from employing SHN-type distributed object servers. Distributed object servers preserve purely objectified representations of information as it moves throughout the system. This is due to the fact that the internal mechanisms of distributed object servers process information as objects themselves.

The IMMACCS model takes full advantage of these object-oriented facilities by integrating an Object-Oriented DBMS (OODBMS) for maintaining persistence. The OODBMS is the facility that the SHN uses to store the application's objects. Employing an OODBMS to store the information objects has two significant advantages.

First, an OODBMS retains the object-oriented representational nature of the information as it exists in its persistent form. Whenever there is representational degradation there is potential for loss of informational content and meaning. By utilizing both transport and storage facilities that are capable of processing and manipulating information as objects, there is virtually no degradation of representation as information flows throughout the system.

The second advantage relates to the manner in which SHN clients request information. Whether mining for information or posting a standing subscription, clients formulate their information requests in terms of objects. More specifically, these subscriptions and queries are formed in terms of object attributes and object relationships, and can range from simple existence criteria to more complex relationships incorporating both logical and relational operators. For example, one such query may request all friendly tracks possessing munitions with an Effective Casualty Radius (ECR) of 500 meters. In this example, the client is essentially pulling information out of the SHN. The operands of the query are each specified in terms of the application's object model.

Another method employed to obtain information from the SHN deals with the notion of subscription. Clients can dynamically register standing subscriptions with the SHN which are again described in terms of the application's object model. For example, a client may request to be notified whenever an enemy track moves to within 300 meters of the client's current location. Once registered, this condition is continuously monitored by the SHN. When satisfied, the SHN essentially pushes the query results to whichever client has indicated an interest (i.e., registered an appropriate subscription). The alternative to this subscription mechanism would be to have interested clients perform the same query on an iterative basis until such a condition occurs. Under these conditions each unsatisfied query may potentially decrease resources (i.e., computing cycles) available to other application components and would essentially prove to be wasteful. If a client takes a more conservative approach where the repeated query is made on a less frequent basis, the client risks being out of date with the current state of affairs until the next iteration is performed. With this in mind, the incorporation of an ability to push information to interested clients becomes essential in providing IMMACCS with an efficient, up-to-date information environment.

The agent engine

The Agent Engine represents the logic-tier of the underlying three-tier architecture of IMMACCS. Existing as a client to the SHN the Agent Engine is capable of both obtaining and injecting information into the SHN. Architecturally, the Agent Engine consists of an agent server capable of serving collections of agents (Fig.2). These collections, or Agent Sessions, exist as self-contained, self-managing agent communities capable of interacting with the SHN to both acquire and inject information. As a SHN client possessing and registering interests in events and information, agent activity is triggered by changes in the battlespace. Regardless of whether agents are interacting with the SHN or each other, interaction takes place in terms of objects. This again illustrates the degree to which an object representation is preserved as information is processed throughout IMMACCS.

The client user interface

Representing the third and final tier of the three-tier architecture employed by IMMACCS the Client User Interface (CUI) exists as a culmination of instances of the 2D-Viewer and the IMMACCS Object Browser. Collectively, these CUIs provide human users with a means of viewing and manipulating the information and analysis provided by the other two tiers of the IMMACCS decision-support system. Understanding the importance of data presentation, the CUIs present the user with this information and analysis in a robust and graphical manner.

As clients of the SHN, CUI users have the ability to interact with each other in a collaborative fashion. That is, by virtue of either injecting or obtaining information from the SHN, CUI users working on the same view have the potential of exchanging strategic or other kinds of information in a collaborative manner. This type of information exchange occurs regardless of whether the relevant view represents the 'common tactical picture' or exists as a localized strategy explored by a subset of users. All information and analysis remains localized within a particular view unless explicitly copied into another view through user interaction. In this manner, no informational or analytical collisions occur between conceptual views without the potential for user-based supervision and subsequent reconciliation.

Implemented agent capabilities

The following agent capabilities are currently supported by IMMACCS and were field-tested during the Urban Warrior Advanced Warfighting Experiment, held in Central California during March 1999 (Fig. 3).

Sentinel Agents: These mentor agents are dynamic and can be created by users and tasked to monitor and alert on simple conditions. For example, Sentinel Agents can be used to generate alerts if an enemy unit or a hostile civilian group enters within the area of a given radius around a location in the battlefield. This agent capability can also be useful for monitoring potential targets of indirect fires, or may be employed by each friendly unit in the battlefield to monitor its immediate surroundings. Once a condition is no longer valid, the agent associated with it can be removed by the operator.

Fires Agent: This service agent capability is static and responds to 'Call for Fire' messages. The agent(s) will select the best weapon that is available, deliverable, and acceptable. During the Urban Warrior AWE the weaponeering portion of this capability addressed range, time of flight, target type, urgency, circular error of probability (CEP), effective casualty radius (ECR), availability, and rules of engagement (ROE). The deconfliction portion of this capability addressed trajectory of munitions relative (i.e., within time and space) to the position of other friendly assets (i.e., people, equipment and other munitions), civilian tracks, and infrastructure objects.

Rules of Engagement (ROE) Agent: This service agent capability is static and monitors for violations in rules of engagement (ROE), in a simplified manner (e.g., entry of tracks into an area of the battlefield that is designated as off-limits). It will also augment the weapons selection and deconfliction capabilities by alerting on available and deliverable weapons that violate the current ROE.

Engagements Agent: This service agent capability is static and monitors incidents of friendly units subjected to enemy fire. Detection of this condition will produce an alert with associated position information. It will also track and alert on sniper fire, terrorist activity, and rioting, highlighting these occurrences on the map.

Logistics Agent: This service agent capability is static and monitors the general readiness of friendly forces. The levels of certain logistics items, such as fuel and water, is monitored with alerts being generated as levels fall below preset thresholds. Upon alert creation the location of potential re-supply points is highlighted on the screen display of the battlefield.

Hazard Agent: This service agent capability is static and monitors the battlespace for indications of nuclear, biological and chemical (NBC) weapons. Upon receipt of NBC indicators, a warning alert is issued indicating the presence and highlighting the suspected position. The friendly units closest to this location are also automatically identified for reconnaissance (RECON) planning purposes.

Intelligence Agent: This service agent capability is static and allows users to monitor enemy sensors. If the sensor is passive the Intelligence Agent sends an alert to propose the initiation of a 'Call for Fire'. However, if the sensor is active the Intelligence Agent will automatically generate a 'Call for Fire'.

Decision Point Agents: This mentor agent capability is a specialization of the Intelligence agent and allows users to request notification as soon as certain user-specified conditions occur within a Named Area of Interest (NAI). For example, a particular road junction may serve as a 'decision point' and alternative courses of action may be attached to the movement direction of the enemy force as it leaves the road junction.

Blue-On-Blue Agent: This service agent capability is static and monitors threats posed by

the proposed actions of one friendly unit to another friendly unit. For example, the Blue-On-Blue Agent will generate alerts if either a friendly unit is directly targeted by the 'Call for Fire' of another friendly unit, or if the likely result of a 'Call for Fire' could indirectly endanger the unit.

Conclusion

IMMACCS is the forerunner of a new wave of decision-support applications that will become widely available to the military during the first decade in the 21st Century. All of these applications will embody at least three fundamental system design concepts. First and foremost, they will incorporate a high level representation of the application domain (i.e., the problem or decision making situation) in terms of objects and relationships among objects. There will be an increasing trend to model the complexities of the problem situation in the ontology (i.e., object model) rather than the agents. In fact, it is likely that the agents in such applications will become relatively simple code modules, with narrowly focused capabilities, but numerous in number.

Second, the ability of these decision-support applications to process 'information' rather than 'data' will allow them to function in a more collaborative role with the human user. Automation will become very much a secondary consideration. The primary concern will be for computer-based agents to assist the human user and each other in collaborative planning, execution and training tasks. In such a collaborative environment emphasis will be placed on the ability of agents to explain their decision process and conclusions, rather than on the automation of the solution generation sequence.

Third, decision-support software will increasingly assume the character of a set of tools. In legacy applications processing 'data' there is no alternative to the incorporation in the software of predefined and hard-coded solutions. This forces the problem in the real world, which is likely to deviate in one or more respects from the anticipated problem, to be distorted to fit the ready made solution that has been built into the software. Software systems like IMMACCS incorporate no solutions, but rather a set of powerful tools. Agents, with their ability to autonomously navigate within the object model representing the problem situation, freely communicate, and dynamically create relationships, constitute the most sophisticated members of this toolkit.

References

[1] R. Leighton, K.J. Pohl, M. Porczak, A. Davis, L. Vempati, H. Assal, and J. Pohl, "IMMACCS After Action Report", CAD Research Center, Cal Poly, San Luis Obispo, CA 93407, Apr. 1999.

[2] L. Myers and J. Pohl, "ICDM: Integrated Cooperative Decision Making – In Practice", *Proc. 6th International Conference on Tools with Artificial Intelligence*, New Orleans, Nov.6-9, 1994.

[3] K.J. Pohl, "ICDM: A Design and Execution Toolkit for Agent-Based, Decision-Support Applications", *in J. Pohl (ed.) Advances in Collaborative Design and Decision-Support Systems*, Focus Symposium, InterSymp-97, Baden-Baden, Germany, Aug.18-22, 1997, pp. 101-110.

[4] J. Pohl, L. Myers, A. Chapman, J. Snyder, H. Chauvet, J. Johnson, and D. Johnson, *ICADS Working Model Version 2 and Future Directions*, Technical Report (CADRU-05-91), CAD Research Center, Cal Poly, San Luis Obispo, CA 93407, Jan. 1991.

[5] J. Pohl, J. La Porta, K.J. Pohl, and J. Snyder, *AEDOT Prototype (1.1): An Implementation of the ICADS Model*, Technical Report (CADRU-07-92), CAD Research Center, Cal Poly, San Luis Obispo, CA 93407, Aug. 1992.

[6] CADRC, "ICODES: Proof-of-Concept System – Final Report", Contract # N47408-93-7347 (Naval Civil Engineering Laboratory), CAD Research Center, Cal Poly, San Luis Obispo, CA 93407, 1994.

[7] K. Penmetcha, A. Chapman, and A. Antelman, "CIAT: Collaborative Infrastructure Assessment Tool", *in J. Pohl (ed.) Advances in Collaborative Design and Decision-Support Systems*, Focus Symposium, InterSymp-97, Baden-Baden, Germany, Aug.18-22, 1997, pp. 83-90.

[8] R. Nadendla and A. Davis, *FEAT: Distributed Problem Solving in a Military Mission Planning Environment*, Master Thesis, Computer Science Department, Cal Poly, San Luis Obispo, CA 93407, 1995.

[9] J. Pohl, M. Porczak, K.J. Pohl, R. Leighton, H. Assal, A. Davis, L. Vempati, and A Wood, *IMMACCS: A Multi-Agent Decision-Support System*, Technical Report (CADRU-12-99), CAD Research Center, Cal Poly, San Luis Obispo, CA 93407, Aug. 1999.

[10] L. Myers, J. Pohl, J. Cotton, J. Snyder, K. Pohl, S. Chien, S. Aly, and T. Rodriguez, *Object Representation and the ICADS-Kernel Design*, Technical Report (CADRU-08-93), CAD Research Center, Cal Poly, San Luis Obispo, CA 93407, Jan. 1993.

[11] M. Minsky, "Why People Think Computers Can't", *AI Magazine*, vol.3(4), Fall 1982.

[12] J. Pohl, "The Representation Problem in CAD Systems: Solution Approaches", *in J. Pohl (ed.) Advances in Cooperative Computer-Assisted Environmental Design Systems*, Focus Symposium, InterSymp-95, Baden-Baden, Germany, Aug.16-20, 1995.

[13] C. Conwell. *Joint Warfare Simulation Object Library*. Naval Command. Control and Ocean Surveillance Center, RDT&E Division. Jun. 1995.

[14] DARPA. *Object Model Working Group Command and Control Schema*. Washington DC. Oct. 1996.

[15] GRC. *The Joint Warfare System Object Model*. Office of the Secretary of Defense. Director for Program Analysis and Evaluation. Joint Warfare System Office. Sep. 1996.

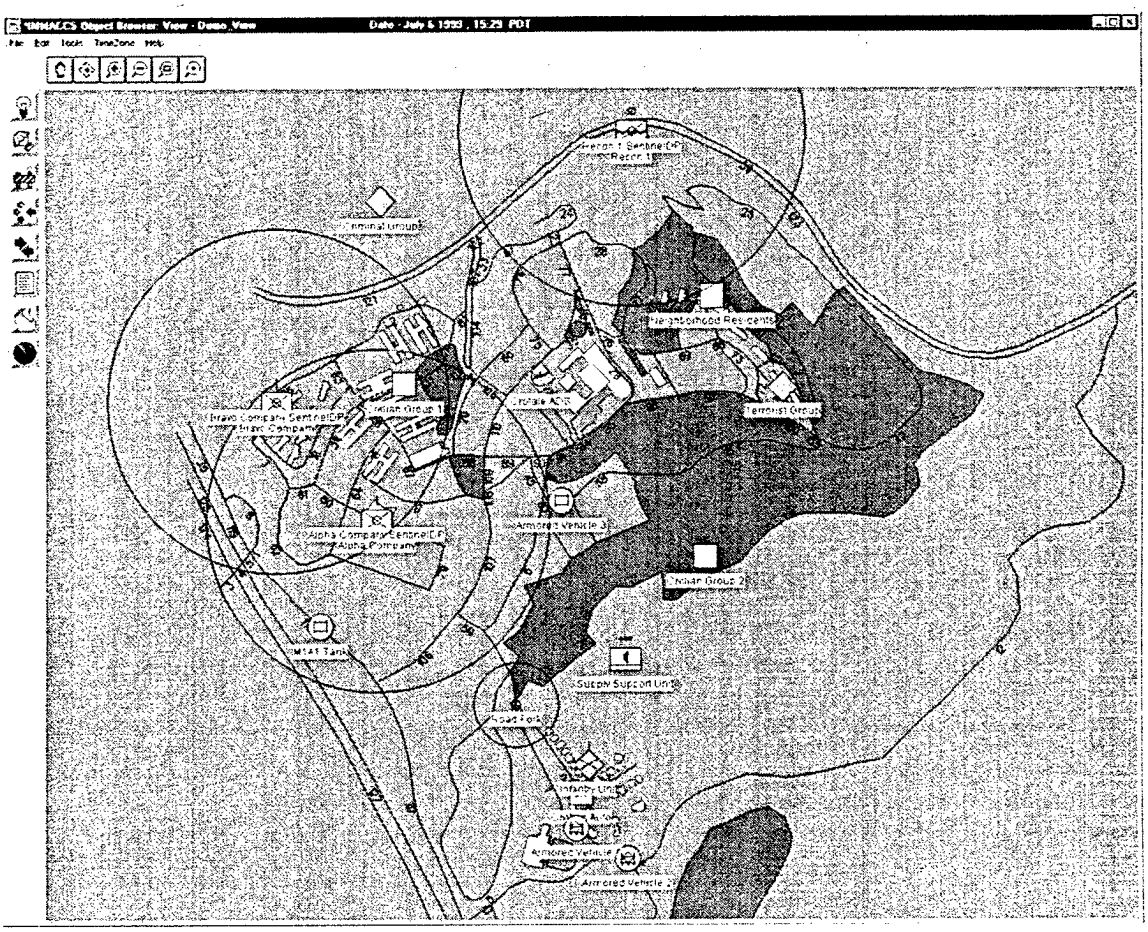


Figure 3. Main screen of the IMMACCS Object Browser user-interface showing a column of agent status windows on the left and the Oak Knoll (Oakland, CA) battlefield area of the Urban Warrior AWE in the central area of the screen.

An Approach, Using Cognitive Engineering and Modeling Techniques, for Realizing Seamless Human Supervisory Control of Complex, Automated Systems and Enterprises

Frank C. Vaughan
Johns Hopkins University Applied Physics Laboratory
E-mail: frank.vaughan@jhuapl.edu

Cory C. Sheffer
Johns Hopkins University Applied Physics Laboratory
E-mail: cory.sheffer@jhuapl.edu

Abstract

With the increasing capabilities of technology to collect, automatically generate, and disseminate information, the challenge is to be able to use this real-time information to re-direct enterprise operations effectively. Mastering this challenge will ultimately require the human, in effecting supervisory control, to operate as a seamlessly embedded control element in the overall process. The still infant but maturing field of cognitive engineering and modeling offers an opportunity for achieving this goal of seamlessly bridging the current gap between fully or semi-automated monitoring and control components and the higher level reasoning capabilities of the human operator/supervisor. A conceptual approach or architecture, employing the use of machine-executable cognitive models of human expertise and decision-making behavior to optimally distribute monitoring, decision-making and control between the fully automated elements and human-directed (or supervisory control) elements in the monitoring and control of complex systems or enterprises is put forward in this paper.

1. Introduction

Monitoring and control of the large-scale dynamic system that comprises the modern enterprise with all of its broadly distributed and potentially conflicting goals, resources and constraints, is critically dependent on the cohesive interactive operation of both human as well as multiple autonomous and semi-autonomous participants of an artificial or machine/computer-based nature (examples of such enterprise-level operations include large military operations, financial/trading institutions, logistics systems, manufacturing plants, power grids, etc.). The ever-increasing capabilities being provided by current-day communications, networking, and information processing technologies to collect, automatically generate, and disseminate information offer the promise of significant improvements in the way that enterprise operations are carried out by automating large segments of the monitoring and control of enterprise processes. This approach will arguably increase the system's ability to use the available information in real time to monitor, control, and re-direct enterprise operations more effectively and more efficiently.

However, the very complex nature of the operations and their interactions that together effect a functioning enterprise dictate that human decision making and judgement will forever remain a critical component of the modern-day enterprise. As one moves up the hierarchy of decision and control in any reasonably complex enterprise, one invariably reaches a level at which current theories and methodologies for automating monitoring and control processes are insufficient. Although this level may continue to shift upward on the complexity scale with continuing advances in control theory and practice (witness the current research in discrete event dynamic systems, for example), there will always remain a set of elements at the top of the decision-making hierarchy that are not amenable to machine implementation and which, therefore, must be addressed by the human decision maker.

Thus being able to make the most effective and efficient use of the ever-increasing amount of real-time information to monitor, control, and re-direct enterprise operations requires the judicious use of automation interacting in a cohesive manner with expert human decision-making and judgement. Optimizing this human-automation interaction will ultimately require the human, in effecting supervisory control, to operate as a seamlessly embedded control element in the overall control loop. However, there is a gap or impedance mismatch between the human decision-maker's higher level reasoning using high level languages and symbologies associated with supervisory control in a specific enterprise domain and the lower-level algorithmic implementation of fully or semi-automated monitoring and control components in the system. Historically and traditionally this gap has been filled by human operators of enterprise subsystems who are intimately familiar with the lower-level automated subsystems and who interact and exchange information with higher level, supervisory control decision-making in a hierarchical command and control structure.

Maintaining this human-intensive hierarchy of information exchange and decision-making yields an enterprise that cannot take full advantage of the increasing availability of real-time monitoring and control information and thus becomes inefficient, falling short of the level of effectiveness that is potentially realizable.

The still infant, but maturing field of cognitive engineering and modeling offers some promise for achieving this goal of seamlessly bridging the current gap between operation of the fully or semi-automated monitoring and control components and the higher level reasoning capabilities of the human operator/supervisor. The basic idea is to ultimately develop a cognitive modeling-based architecture that, in some sense, optimally distributes decision-making and control between the fully automated elements and the human-directed (i.e., supervisory control) elements in the monitoring and control of complex systems. The motivation for taking this approach is the thesis that the keys to achieving any significant level of success in automating complex systems are:

1. A reasonably powerful ability to emulate expert human reasoning/decision-making at a useful level of complexity in hardware/software,
2. The capability for efficiently and effectively providing the right information at the right time to human supervisors in a mode that is relevant to the task at hand in order to effect responsive human supervisory control when and where needed, and
3. The ability to monitor and control discrete event-driven processes that are non-continuous in time and inherently non-linear in nature.

Cognitive engineering and modeling addresses elements of all three of the above key requirements. The first two are, of course, complementary in that the relative distribution of workload between the "pure automation" components and the human operator/supervisor depends on how successful we are in emulating human reasoning in automated components. If we could successfully develop an architecture that provides an implementation framework that is common to all three of the above elements, then the methodology should be extensible to virtually all automated systems applications and can easily adapt to increasing technological maturity in any and all of the three areas.

A conceptual approach or architecture, employing the use of machine-executable cognitive agents that contain cognitive models of human expertise and decision-making behavior is put forward and discussed in the last section of the paper. We first provide some background in cognitive modeling by reviewing the salient findings from two applications in which JHU/APL developers employed cognitive modeling to address some of the issues above (Section 2) and we review some key concepts related to supervisory control (first part of Section 3). Taken together, this background material sets the stage for presenting and discussing the notional, cognitive modeling-based enterprise control architecture presented in the last part of Section 3.

2. Cognitive Modeling-Background

To date, our work in cognitive modeling and engineering at JHU/APL has primarily been in the area of user and performance-centered design, conducting development and evaluation of Human Computer Interfaces (HCI). Modeling user decision-making tasks has become the prerequisite step in our development of user interface technologies and prototype development efforts. We use cognitive modeling to capture and represent the goals and actions of users, and to assess the utility of user interface options [1]. Examples of this work include the Tactical Scene Operator/Associate (TSO/A) Program sponsored by the Tactical Technology Office of the Defense Advanced Research Projects Agency (DARPA) and the Intelligent Systems Interface (ISI) program sponsored by the Office of Naval Research (ONR).

The primary domain of concern within the TSO/A Program comprises a critical subset of the overall submarine command and control (C2) process. In that domain, tactical decision makers are concerned with detecting, locating, identifying and engaging hostile submarines or ships while maintaining ownship safety and minimizing the likelihood of counterdetection [2]. The typical process involves the submarine's commanding officer (CO) or officer of the deck (OOD) interacting with multiple human operators of the various sensor and combat subsystems, as well as directly with system displays and information, to maintain awareness of the evolving tactical scene, make decisions regarding what actions to take, and give orders to implement those actions. Thus, the scene assessment process is, in essence, a lower-level enterprise within the larger enterprise that is the submarine with the CO or OOD effecting supervisory control of both levels of enterprise (Figure 1).

Given the austere nature of submarine sonar data and the considerable uncertainty inherent in undersea warfare, much of the interaction of these users with the sensor and combat systems on the submarine is concerned with understanding sensor data and information processing, and gaining confidence in the conclusions arising from processing systems. Consequently, these decision-makers spend a considerable amount of their time investigating the results being presented to them. Selecting the best display format is a critical issue. As a result, understanding the relationship between display formats and user performance is critical, especially when uncertain

and ambiguous information must be presented and understood by the user.

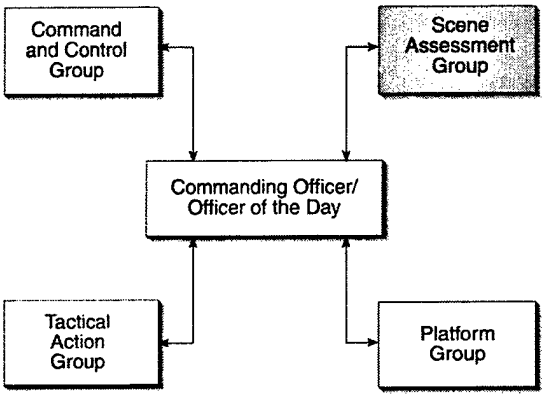


Figure 1. Each watchstation group makes up a lower-level enterprise within the overall submarine operations enterprise [2]

In TSO/A, a top-down design process (Figure 2) was employed to develop a user interface design and generate both the operational and functional requirements for the system [2]. At the top of this process is the identification of the operational requirements. For TSO/A, these consisted of consolidating all tactical scene generation functions into a common watchstation interface and leveraging automation to decrease watchstander workload and improve tactical decision-making. The second step was to perform a functional decomposition of the tactical scene generation task.

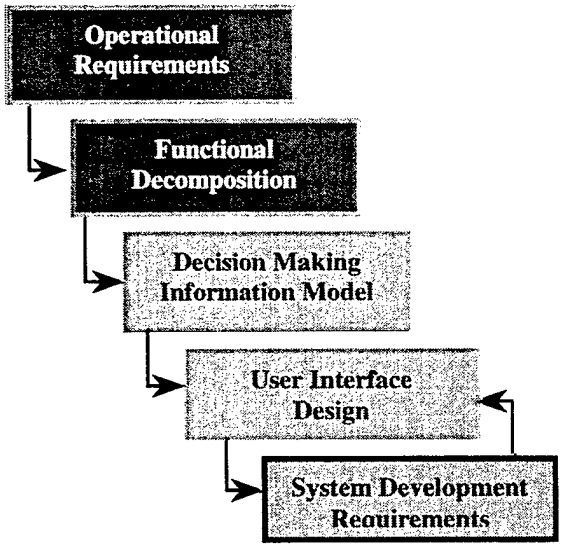


Figure 2 TSO/A Top-Down System Design Approach

The third step, constructing a decision-making information model, was where the majority of our cognitive engineering and modeling effort within the TSO/A Program was focused. In developing a decision-making information model, it is necessary to identify the high level goals of the user and, from there, identify the sub-goals and methods to achieve those goals. This is an extremely important step as it leads to many other pieces of the process and, therefore, careful identification and selection of goals and subgoals is critical in maintaining fidelity of the resulting cognitive model. By combining the results of the decision-making model step with the results of the functional decomposition, a TSO/A user interface design was created; cognitive models were developed to assess the utility of various options in that design; and tasks which required Human-in-the-Loop interaction were identified. The models were then used to assess the different user interface options, allowing an iterative approach in the design of the user interface [2]. This multi-step process is shown in Figure 3.

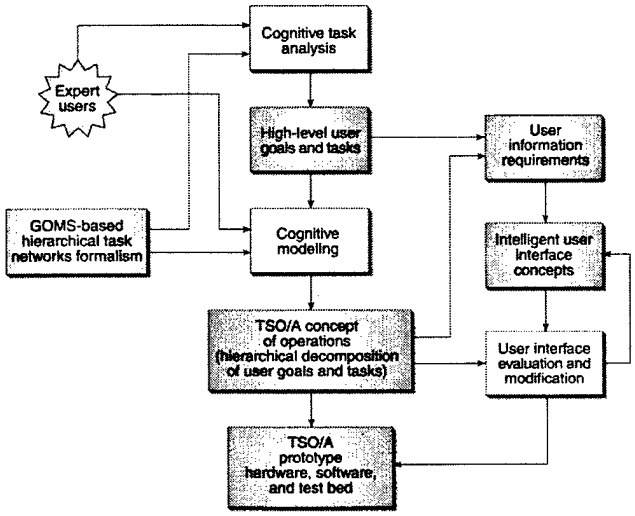


Figure 3. Cognitive task analysis and cognitive modeling for development of TSO/A concept of operations and user interface evaluation [2]

The development of the decision-making information model used a modeling process referred to as GOMS which stands for Goals, Operators, Methods, Selection. The GOMS modeling methodology is based on what is referred to as the Rationality Principle [3], which states that users act rationally to attain their goals. So to predict a user's behavior, one must analyze the task to determine the user's goals and operators and the constraints of the task. This defines the users cognitive structure as defined by the GOMS model. The structure consists of four components: (1) a set of Goals, (2) a set of Operators, (3)

a set of Methods (each composed of a set of operators) for achieving the goals, and (4) a set of Selection rules for choosing among competing methods for common goals.

In both TSO/A and ISI, certain user interface components were evaluated using GOMS models to assess various implementations of the same component. In TSO/A one of these components conveyed pertinent information about the different tracks being presented on the tactical display. In their work, Coury and Strauss [1] examined two different options to accomplish this: 1) a textual format which presented the textual information for each track under major categories of concern to the decision maker, and 2) a matrix format which listed tracks on the left and major categories of concern across the top with a highlighted cell to indicate information present for that track in that category. Coury and Strauss also investigated two ways to visualize the information space in support of the ISI program. One option used three-dimensional cubes to represent various nodes of information in a tree like structure branching down from a given track. Each face of the cube presented different pieces of information for that node. The other implementation was similar, but implemented in two dimensions. Coury and Strauss created GOMS models in both of these cases to examine the user interface value.

Another natural outcome of a carefully implemented decision-maker information modeling process is the identification of which tasks can be automated and which should include human-in-the-loop decisions. In TSO/A, for example, numerous systems are required to provide the supervising officer with the information he needs to accomplish his desired goals in generating and assessing the tactical scene. So many systems are essential that the knowledge necessary to operate the entire collection of systems would call for a "super-operator" with a tremendous amount of knowledge about each individual system. Traditionally, this situation is handled by having individual human operators for each system who interact with the supervising officer to provide him with the distilled information he needs. Thus the equivalent approach for TSO/A is to design a user interface that is simultaneously comprehensive in system coverage yet sufficiently simplified in presentation content to allow operators to concentrate more on information being presented to them rather than the manipulation of systems to view the information they want [2]. To accomplish this, it was necessary during the merging of the various functionalities with the decision maker's goals to identify which functions should be automated, and which should be left up to the decision maker. Through the use of task analysis and the involvement of subject matter experts, certain portions of the tasks were identified as requiring human-in-the-loop interaction. This allowed the computer to take care of what it could do best and get the human involved when the level of information required it. This decreased the workload on the operator, yet did not create a mistrust of the automation.

The primary tool that we employed within the TSO/A Program to accomplish the cognitive modeling is a commercial toolset called COGNET/iGEN™ that is produced by CHI Systems, Inc. COGNET is a hybrid approach in the area of cognitive modeling [4]. It combines both a multi-panel blackboard representing the underlying problem knowledge with multiple tasks representing chunks of procedural knowledge. The structure of these chunks of procedural knowledge is strongly based on Card, Moran, and Newell's GOMS notation [3] with additional features to allow the task to interact with the multi-panel blackboard. CHI System's iGEN™ software implements the COGNET framework in such a way as to allow the generation of intelligent agents which are capable of working in real time and act in a very human-like way. These cognitive agents can then act in real time as both Decision Support Systems and Electronic Performance Support Systems [4]. They can also be used to provide varying degrees of automation. Based on our work in cognitive modeling and integrating human interaction with automation in the task of tactical scene generation, we believe that it is possible to use a modeling framework such as COGNET to create intelligent agents which function as additional operators, creating a seamless interface or bridge between the automated components of a complex system or enterprise and the human operators who are providing supervisory control of the enterprise. This conclusion forms the basis of the notional architecture described in Section 3.

3. Notional Cognitive Architecture for Supervisory Control

In order to set the stage for describing how cognitive modeling can be applied to complement the human supervisor's monitoring, decision-making, and executing control actions in effecting supervisory control of a complex enterprise, it is useful to first review and discuss some tenets and background of supervisory control in general. In developing our conceptual approach of applying real-time, machine-implemented cognitive models of human expertise and decision-making behavior to the process of supervisory control, we found the functional description of supervisory control put forth by Sheridan [5,6] to be a particularly useful description of that process. The utility of Sheridan's functional model is primarily due to two reasons: first, the natural manner in which the constituent components of a COGNET/iGEN-type cognitive modeling architecture, coupled with other automated decision-aiding techniques, overlay the functional components of supervisory control described by Sheridan [6]; and, second, the usefulness of the supervisory control perspective in classifying human

functions with respect to computer functions and the concomitant manner in which computers can aid human operators in implementing supervisory control, in particular [5]. Thus, we first provide a brief summary of Sheridan's functional description of supervisory control.

In the context of the present discussion, i.e., a human operator controlling a complex system or enterprise, the most useful paradigm for supervisory control is the simplified version shown in Figure 4 [6]. The human operator interfaces directly with a human-interactive computer (HIC) or computing module which is, in turn, interfaced to a number of possibly (and, certainly for a large enterprise, most likely) remote computers that are applying some level of automation to monitor and control lower level tasks. These task-interactive computers (TIC) are simultaneously receiving higher-level monitoring and control instructions from the HIC and passing higher-level monitoring information back up to the HIC.

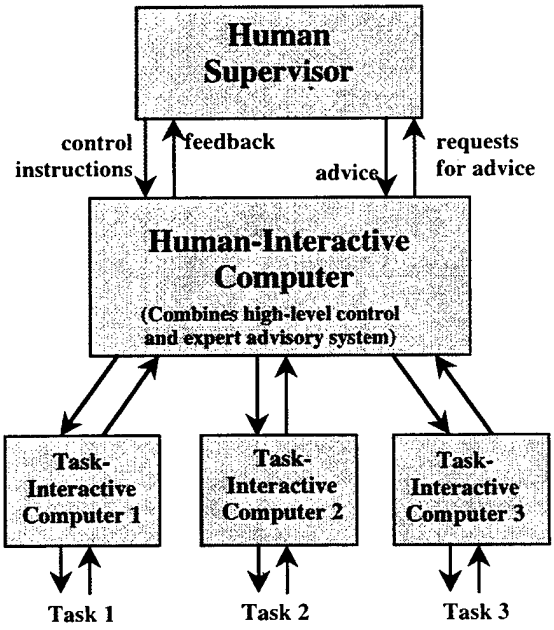


Figure 4. Simplified paradigm for supervisory control of multiple tasks [6]

The human operator communicates with the human-interactive computer using symbolic commands specified in terms descriptive of the particular enterprise domain (i.e. some form of natural language or symbology easily translated by the human operator into terms natural to the enterprise domain). These commands are then translated by the HIC into control instructions for the TIC's, specified in communications and applications protocols that are specific to the task being controlled. Monitoring information, collected as part of a task operation, is successively passed back up, via the TIC and HIC, and is either acted on directly by the HIC (which may result in control instructions being immediately passed back down to the TIC) or passed up to the human operator, again after being translated by the HIC into enterprise domain symbology readable by the human operator. Although in a real enterprise there will most likely be multiple levels of abstraction between the lowest level tasks and the highest level human supervisor (and some of these levels may involve a human operating as a lower-level supervisory controller), this simplified paradigm is sufficient to define and discuss the basic functions or operations that must be carried out by the human supervisor in implementing supervisory control.

The five generic supervisory functions discussed by Sheridan [5,6] are:

- Planning what task to do and how to do it
- Teaching (or instructing) the computer (i.e., HIC) what was planned
- Monitoring the automatic action and results of automated monitoring and control to ascertain that overall system/enterprise is operating as planned
- Intervening (i.e., the human supervisor supplements ongoing automatic control activities, takes over control entirely after the desired goal state has been reached satisfactorily, or interrupts the automatic control in emergencies to specify a new goal state and reprogram a new procedure)
- Learning from experience in order to improve the overall enterprise operation

With the exception of planning, these functions are self-explanatory at a top level (Sheridan provides a more detailed description of each of these functions in both of the references [5,6]). It is informative and useful, however, to break down the planning function further. It includes having (or gaining, if necessary) an understanding of the system or enterprise being controlled; setting goals or objectives that are attainable along with tradeoffs that the HIC can "understand" in order to be able to provide advice or make control decisions itself; and formulating a strategy for going from the initial state to the goal state. Notice the emphasis on goals or objectives as specifying the desired state of the enterprise or system. The fundamental nature of supervisory control is that the human supervisor does not provide direct control signals to lower-level subsystems (with the possible exception of the intervene function) but instead specifies the desired state of the enterprise in terms of goals or objectives specified in the terminology or symbology of the application domain. The monitoring function then consists of comparing the evolving state of the enterprise to the desired goal state established in the planning phase. This goal-directed nature of supervisory control is one of

the key aspects that make it a prime candidate for GOMS-type task analysis and associated cognitive modeling.

A useful way to view the actual implementation of these generic supervisory functions as executed by the human supervisor in the continuous operation of the enterprise is that they operate as a set of three nested control loops as depicted in Figure 5 [6]. The innermost loop, in which the monitoring function closes on itself, involves the human supervisor continuously assessing the goal state of the enterprise based on observations of appropriate enterprise conditions and providing minor system adjustment inputs, via the HIC, into the appropriate TIC's. No significant intervention is required. The middle loop shows the intervening function looping back to the teaching function. This loop is initiated when the monitoring loop indicates that either the enterprise state is significantly deviant from the desired goal state as to require specification of a new, perhaps intermediate goal or a new goal state has resulted from replanning. In either case, a new goal state has resulted and the teaching function must be activated to instruct the HIC to reprogram for the new goal state. The outer loop closes from learning back to planning indicating that the human supervisor has learned some new information about the enterprise that he/she did not have previously that is significant enough as to require modifications in the top-level planning of goals and objectives. A good example of the latter control loop, taken from the tactical situation assessment domain as described in the discussion of the TSO/A Program above, might be the sonar and combat systems operators discovering an improved method for setting up system parameters that provides better performance against a particular type of contact.

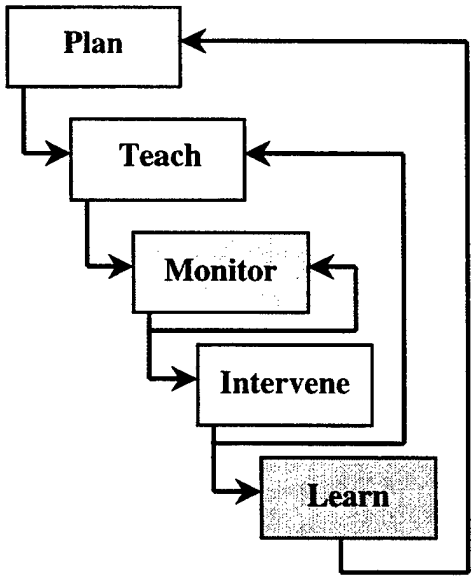


Figure 5. Five supervisory functions as nested loops [6]

In assessing where real-time executable cognitive agents may be applied to assist and complement the human supervisor in carrying out the five generic supervisory functions, it is very instructive to look at these functional aspects of human behavior in terms of the level and type of mental activity involved. One model for such characterization that is well known in the area of cognitive psychology is that put forth by Rasmussen [7] which introduced a paradigm that breaks human behavior into three levels. The lowest level in this three-tiered hierarchy is called skill-based behavior and involves continuous, typically well-learned sensorimotor type behavior. The next higher level is designated as rule-based behavior and is characterized as decision-making that can be represented by a set of production rules of the "if-then" type of formulation. The highest level is that of knowledge-based behavior. Examples of the latter include such mental activities as high-level situation assessment and evaluation, evidence-driven alternative hypothesis formulation, etc., that are characterized as drawing on sets of knowledge, experience, and expertise that have been developed over time and are retained in a person's long-term memory (the rules involved in rule-based behavior are, of course, also retained in long-term cognitive memory whereas skill-based behavior is arguably retained in a lower-level, non-cognitive type of memory).

If we now look at the functions involved in each of the three supervisory control loops described above, we can characterize each loop as being associated with one of the levels of Rasmussen's hierarchy of human behavior. Sheridan [6] makes the argument that the functions/loops can be allocated in the following manner. System monitoring, characterized as involving "semi-continuous, well-learned, and largely perceptual-motor skills", fits the Rasmussen skill-based behavior level. The loop involving teaching and intervening employs "computer instructions and task protocols initiated by established criteria, and therefore tend to be rule-based." Since the loop involving planning and learning require "attention to goals and problem formulation," these functions are more knowledge-based in nature.

Now, recall that the cognitive tasks that are implemented in the COGNET/iGEN formulation contain the procedural knowledge which, along with the problem (or system/enterprise state) knowledge contained in the blackboard memory are employed to emulate expert human decision-making appropriate to the domain of interest and the current and desired enterprise goal states. Thus, to the extent that we can successfully capture the expert human decision-making procedures with the GOMS-type approach to task modeling, we can implement high-level rule-based as well as lower-level knowledge-based behavior in the HIC in the form of real-time executable iGEN cognitive agents. If we assume that the monitoring function, being a skill-based behavior, is amenable to lower-level automation (i.e., non-cognitive),

this function can be implemented entirely within the TIC's. In fact, this level of monitoring is essentially transparent to the human supervisor. The monitoring information that the supervisor receives will, most likely, have been abstracted by cognitive agents in the HIC from a collection of lower-level monitoring inputs that the TIC's have provided to the HIC.

By implementing, in the form of real-time executable cognitive agents, those supervisory functions involving a combination of lower-level knowledge-based reasoning and rule-based decision-making that are amenable to GOMS-type task analysis and associated cognitive modeling, we will have, in essence, created a common framework of shared procedural knowledge and expertise between the human supervisor and the HIC as depicted in Figure 6. Thus we will have succeeded in distributing the supervisory control loop functions, in a virtually continuous and seamless manner, between the computer and the human supervisor. Since the fully automated and semi-automated components of the overall enterprise control loop are already presumably implemented among the TIC's and are, therefore, by virtue of the computer-to-computer interface between the HIC and the TIC's, already seamlessly interfaced with functionality resident in the HIC, we will have succeeded in seamlessly embedding the entire supervisory control loop, including the human supervisor's functions, in the overall enterprise control loop.

We can further unify the configurational architecture by taking advantage of the services provided by the COGNET/iGEN run-time system to incorporate the functions that would have been executed within the TIC's into the COGNET/iGEN run-time configuration. With the HIC functions already implemented in this configuration, the COGNET/iGEN blackboard structure and associated messaging protocols provide a natural interface between the functionalities that had been resident in the HIC and the TIC. Since, in this architecture, the functions of both the HIC and the TIC would be implemented in the form of "cognitive agents" (albeit relatively simple ones for some of the TIC functions, for example), this approach is tantamount to having sets of Human Interactive Cognitive Agents (HICA) and Task Interactive Cognitive Agents (TICA) replace the HIC and the TIC, respectively.

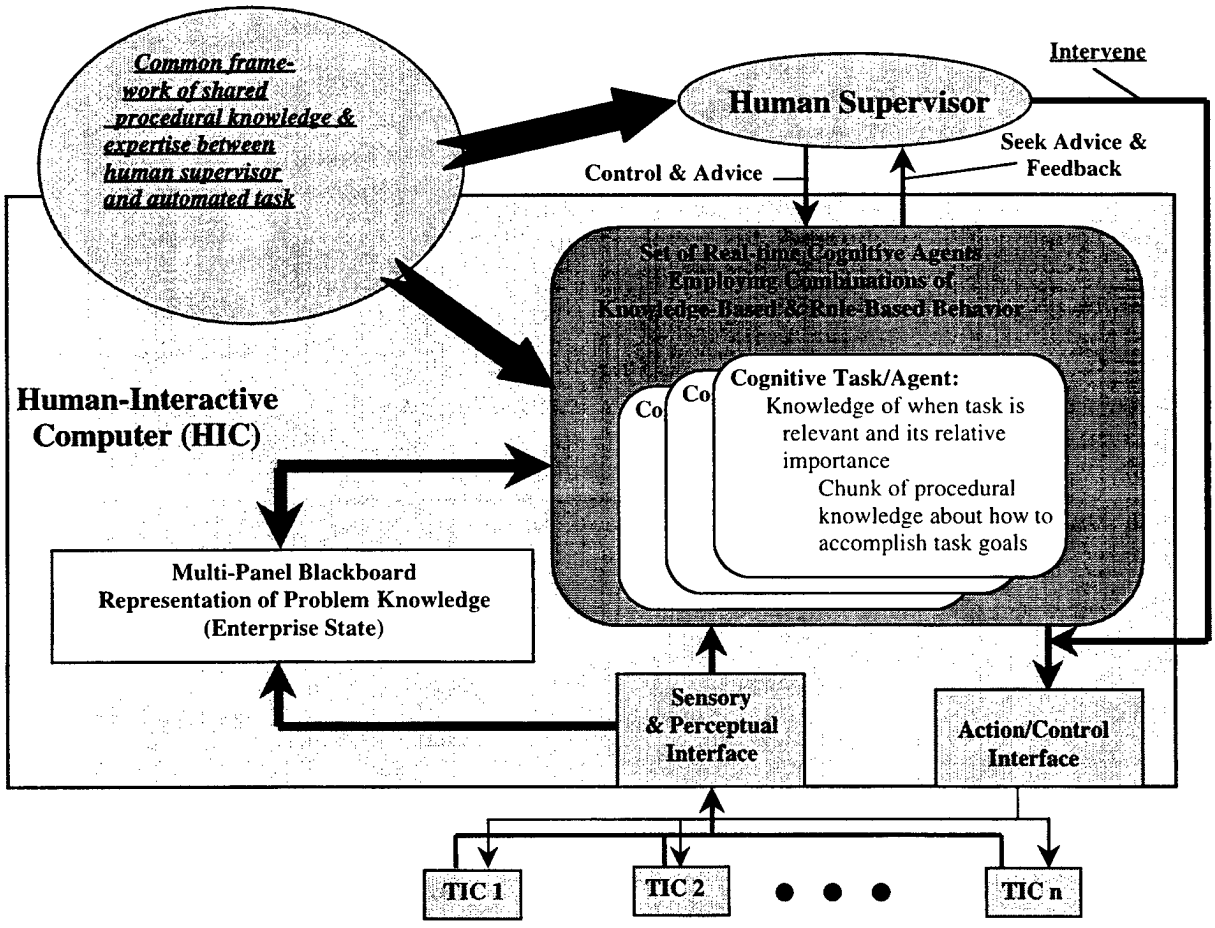


Figure 6 Use of cognitive modeling to develop and implement common framework of shared procedural knowledge between human supervisor and human interactive computer

Such a distributed architecture is depicted in Figure 7. Although perhaps requiring some redesign of the blackboard implementation configuration and messaging protocols, such a functionally distributed approach should easily be able to be implemented over a network of distributed components. It should, therefore, be able to support enterprises with geographically widely distributed elements. Because of the nature of its very modular separation of problem (or enterprise goal state) knowledge and procedural knowledge this architecture should be easily expandable and extensible and easily adaptable to increasing technological maturity in both control system automation and automated reasoning techniques. Thus, by creating the possibility of this common framework of procedural knowledge that is shared between the human supervisor and the computer, the cognitive modeling and executable cognitive agent structure of a cognitive modeling and run-time environment like that provided by COGNET/iGEN offers the promise of realizing truly seamless human supervisory control of complex, automated systems and enterprises.

Of course, such an approach remains notional at this point until some of the more challenging technical obstacles are addressed experimentally. Perhaps the most significant of these is the difficulty of modeling cognition or mental activity in general. The process of identifying goals and decomposing those goals into operational tasks, in a form amenable to machine execution, by observing and interviewing human experts is a very difficult one. In particular, the relative role that subjective judgement plays in supervisory control in setting objectives, command/control decision-making, etc. as a function of enterprise domain is very important in both deciding on an overall approach for goal-setting and task decomposition as well as determining the expected limits on the cognitive models which result. However, because of the relative promise in the cognitive engineering and modeling approach in being able to emulate expert human decision-making in automated systems, continued research in this area should be of high priority.

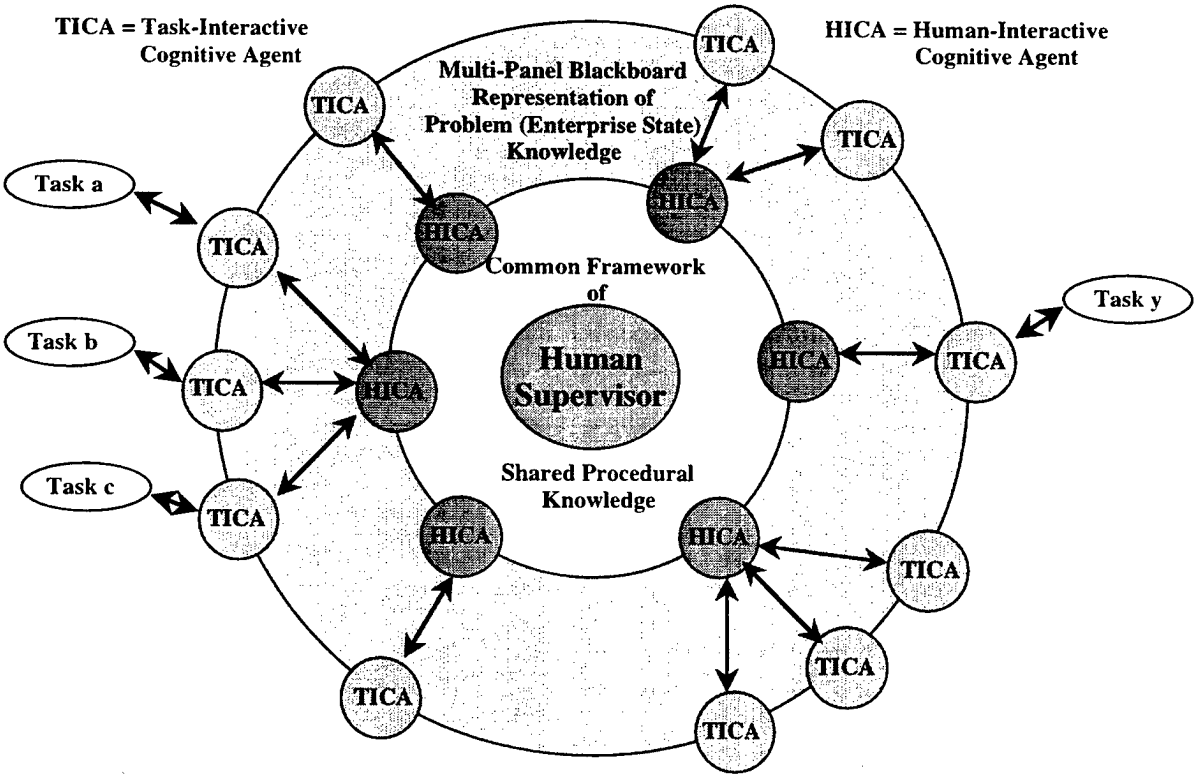


Figure 7 Distributed, cognitive agent-based architecture for seamless human supervisory control of complex automated systems and enterprises

References:

[1] B. G. Coury and R. A. Strauss, "Cognitive Models in User Interface Design," *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting*, October 1998, in press

[2] C. C. Sheffer and F. C. Vaughan, "Application of Cognitive Modeling to Tactical Scene Generation," *Johns Hopkins Technical Digest, Volume 20, Number 3*, Johns Hopkins University Applied Physics Laboratory, Laurel, MD, September 1999, pp. 253-261

[3] Card, S. K., T. P. Moran, and A. L. Newell, *The Psychology of Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum, 1983

[4] Zachary, W., J-C. Le Mentec, and J. Ryder, *"Interface Agents in Complex Systems,"* in C. Nutuen and E. H. Parks (Eds.), *Human Interaction with Complex Systems: Conceptual Principles and Design Practice*, Norwell, MA: Kluwer, pp. 35-52

[5] Sheridan, T. B., *"Task Allocation and Supervisory Control,"* in M. Helander (Ed.), *Handbook of Human-Computer Interaction*, North-Holland, Amsterdam, 1988, pp. 159-173

[6] Sheridan, T. B., *"Telerobotics, Automation, and Human Supervisory Control,"* MIT Press, Cambridge, MA, 1992

[7] Rasmussen, J., *"Outlines of a Hybrid Model of the Process Plant Operator,"* in Sheridan, T. B. and G. Johannsen, (Eds.), *Monitoring Behavior and Supervisory Control*, Plenum, 1976

Deriving Importance of System Capabilities from Mission Success Utilities

Dr. Paul E. Girard
Science Applications International Corporation
4015 Hancock Street, M/S T-1
San Diego, CA, 92110
(858) 826-7158, FAX: (858) 826-5617
paul.e.girard@saic.com

Dr. Marco Fiorello
ACS Services
1550 Hotel Circle N., Ste. 200
San Diego, CA, 92108
(619) 718-9600, FAX: (619) 718-9611
marco.fiorello@acssd.com

Abstract

The authors have considered an analysis approach that uses Multi-Attribute Utility Theory (MAUT), a rational, formal decision theory, to express measures of performance (MOP), importance (MOI), and effectiveness (MOE) in a hierarchy of objectives and purposes, related to mission objectives, operational procedures, and system capabilities. This approach is the first to provide a morpho-logical basis for distinguishing between MOPs and MOEs. A newly derived measure of importance (MOI) provides a tie between MOPs and MOEs that is an innovative and essential element of analysis.

Systems are described in terms of their performance of certain characteristic functions which that type of system performs. For example, sensors and weapons are judged by their probabilities of detection and kill. In order to assess C3 systems, we must be able to model how likely the decision maker is to recognize the situation, how likely that person is to choose particular courses of action as a result of that recognition and when these events will take place. The problem of assessing system ***effectiveness*** *is in being able to relate the systems'* ***performance*** *to the mission objectives and collateral outcomes, i.e., what is the* ***importance*** *of the system's capabilities? This issue will always be contextually dependent. Our treatment of measures will address performance, importance, and effectiveness, as well as the role of context.*

This work is based on [6].

1. Introduction

The objective of the JFACC experimentation process is to assess the contribution to the JFAC mission from candidate changes in organization structure and processes and from candidate new technologies. At the top-level there is a need for an evaluation framework that meets the following criteria:

(a) Allows a correct and explicit reflection of JFAC mission objectives
(b) Discriminates effectively and equitably across the alternatives to be evaluated
(c) Incorporates a reliable and repeatable procedure
(d) Is understandable to decision makers and domain experts
(e) Incorporates a full breadth of measures of contribution including preferentially independent aspects of value.

One approach that meets these criteria is multi-attribute utility theory (MAUT). An MAU function is expressed as a preference structure (importance weighting) on the aspects of the JFAC mission purpose, which, when combined with the degree of achievement of those purposes, yields an expected effectiveness, which can be used to rank alternative solutions (means). The degree of achievement is quantified in terms of likelihood/probability of operational outcomes with respect to those objectives. But reaching these objectives requires contributions from systems and procedures with lower-level purposes.

There is an underlying hierarchy of objectives [1]-[3], whereby each purpose is supported by sub-purposes and functions and tasks to accomplish them. Figure (1) portrays a small portion of a JFAC hierarchy of objectives, ranging from Mission-objectives (ends) and the alternative ways (means and methods) being considered for achieving or contributing to the mission. The MAUT approach, using relative weights (importance) and degree of achievement (probabilities) of the sub-purposes, might be applied at the lower level. It is preferable that this be done at the lowest parameter level in terms of class intervals that span the range of distinguishable or meaningful differences among the alternatives. It is also usually easier to elicit knowledge, conduct experiments or develop models at that level.

When completed, the hierarchy will include weights for each sub-purpose to reflect functional importance relative to the mission objectives, and at the lowest level it will incorporate class intervals to reflect levels of achievement.

To be correct, the MAU function must only apply to the top-level mission-objectives. To be consistent, the effectiveness attributable to the importance and achievement of those lower level purposes associated with the alternative means being considered, should

match that due to the importance and achievement of the top-level objectives. The Derived Utility approach described below is intended to provide that consistency.

Because the importance of objectives and therefore, of system capabilities varies with context, different scenario settings will be used to determine the stability or sensitivity in the importance weights. Each candidate alternative experiment under consideration will then be evaluated in terms of its contribution to the JFAC mission as define in the MAU hierarchy.

2. A Previous Mathematical Formulation

A previous workshop [4] promoted a mathematical formulation of effectiveness based on a mapping or model of the probability of success at the mission level given performance at the lower level. The combat model conditioned a set, R, of combat outcome (results) state vectors, $\underline{r}$, on the values of the vector, $\underline{v}$, of Variables For Measuring Effectiveness (VFME) of C3 systems. The measure was expressed as a conditional probability, i.e., $\mu (R^* | \underline{v}, s)$, where s is a system from the set of system alternatives under consideration. The mission success criterion is to have outcomes occur in R* and that the force should be able to achieve success (in the context of the mission scenario, including assumptions about the enemy) with some percentage (say, T) as a confidence, such as 95%. The requirement set was then defined to be a set, V*, of acceptable outcomes in V, such that

$$V^* = \{ \underline{v} : \mu (R^* | \underline{v}, s) \varepsilon T \} \quad (1)$$

Effectiveness of a given system, s, was expressed as

$$E(s) = \sum_{\underline{v} \in V^*} \mu (R^* | \underline{v}, s) \, dF(\underline{v}, s) \quad (2)$$

(using Σ as either a summation or an integral sign).

where $F(\underline{v}, s)$ is the probability distribution of the VFMEs for the system, s. The last equation is equivalent, conceptually, to

$$E(s) = \sum_{\underline{v} \in V^*} \mu (R^* | \underline{v}, s) \, p (\underline{v} | s) \, d\underline{v}. = \mu (R^* | s), \quad (3)$$

where the latter equality was added by the author, as an extension of the notation used before. This is essentially the probability that system, s, will achieve performance in V*, AND that the force success will therefor be in R*. The measure of system performance is $F(\underline{v}, s)$ or, equivalently, $p (\underline{v} | s)$. The measure of effectiveness was E(s), determined by weighting the system performance by the combat outcomes model, $\mu (R^* | \underline{v}, s)$, and integrating over the successful outcomes.

The above formulation was augmented with discussion of the role of the environment as a conditioning domain, which implicitly included the enemy forces and own forces. There was also a recognition that the performance variables, $\underline{v}$, of systems, s, were dependent on the parameters of the system design, such as antenna aperture and output power.

Some deficiencies of the approach are that it used regions with crisp boundaries to define sets, such as success, and that the definition of performance measures and effectiveness measures was only implicit. The distinction between performance and effectiveness was that the former was a (conditional) probability distribution over a domain of the consequent variables, while the latter was the (conditional) probability of a set in the domain. That the sets were crisp sets means that outcomes outside of R*, which may have had worth to the decision maker were not included, and outcomes outside of V*, which may have resulted in some good outcomes, but not with confidence T, were also excluded. The concept of derived utility will address this issue.

3. Derived Utility Approach to Measures

The authors have considered an analysis approach [5]-[6], that uses Multi-Attribute Utility Theory (MAUT), a rational, formal decision theory, to express measures of performance (MOP), importance (MOI), and effectiveness (MOE). In this theory, utilities are weighting functions defined on the consequent domains of attributes of interest, which are the domains of the top-level objectives in a consideration of a set of alternative courses of action. The choice among C3 systems might be the set of alternatives for an acquisition manager, while, for the operational commander, those choices might be the tactics or organizational structure.

3.1 Utility Theory

Utility Theory is used to describe the motivation for decision making. Given a set of choices (alternative actions or systems to procure), certain outcomes will result, depending on the decision to choose one or another of the alternatives. These outcomes are the set of possible values of attributes or states of the "system" or "situation" with which the decision maker is concerned. These attributes may include physical states and decision (information/choice) states within the operational problem, and some other variables, (e.g., cost, schedule) of the fiscal problem. Some subset of these are called attributes for the particular decision to be made. Utility Theory applies a measure of "worth" to each value of a single attribute (or vector of values of a multi-dimensional problem). For each choice of alternative course of action, there is a probability distribution believed to hold with regard to which value of the attribute(s) will be achieved. The expected "utility" of choosing that alternative is the sum of the

products of the "worth" of an outcome value times the probability of that outcome, given that that alternative is chosen. The expected utility is:

$$EU(A) = \sum u(x) \cdot p(x \mid A)\, dx \qquad (6)$$

where A is a set of alternatives, X is an event (or quality or state) or attribute vector, u(x) is a utility function concerning the attributes,and p(x | A) is a probability of occurrence of particular outcomes of the attribute that depends on the choice of the alternative. The alternative with the highest expected utility should be the "best" choice. This is the fundamental principle upon which Utility Theory is based, that the decision maker should choose the alternative which maximizes expected utility. Experiments show that, while people's utilities can be elicited, their behavior does not necessarily follow the predicted outcome. Thus Utility Theory is known as prescriptive or normative, while theories which represent how decisions are actually made are called descriptive. For our purposes, though, the objective is to find measures, which formalize the concept of effectiveness in terms of worth.

3.2 Proxy Attributes and Derived Utility

When attributes and their utilities are assessed, it may be the case that the attributes are actually proxies for higher level attributes. For example, the attributes of income and net worth may be proxies for the comfort or peace of mind that accrue from that condition. The concept of proxy attributes is used to justify eliciting utility and probability on the proxy attributes, since they tend to be more quantitive or concrete. But, the relationship between the utility of the higher attributes and that of the proxies should depend on the causal relationship of how the latter contribute to the former. While this may not be identifiable for abstract higher level attributes, the concept of proxy attributes will be used to derive such a relationship in the case of quantifiable operational objectives with respect to the capabilities of the systems.

3.3 Derived Utility

When a relationship (model) of the causal conditional effect of the proxy attribute, y, on the higher level attribute, x, is known, a relationship between the utility functions can be found. The expected utility can be expanded to include the model relationship:

$$EU_X(A) = \sum u_X(x) \cdot p(x \mid A)\, dx \qquad (7)$$
$$= \sum u_X(x) \cdot \left[\sum p(x \mid y, A) \cdot p(y \mid A)\, dy\right] dx$$
$$= \sum \left[\sum u_X(x) \cdot p(x \mid y, A)\, dx\right] \cdot p(y \mid A)\, dy$$

where $u_X(x)$ is the utility on x, elicited as worth on the high level attributes (TLWRs), p(x | y, A) is the model of the relationship of outcomes, x, conditioned on capabilities, y, and p(y | A) is the capability expression for alternative A. For the set of proxy attributes, y, the expected utility is:

$$EU_Y(A) = \sum u_Y(y) \cdot p(y \mid A)\, dy \qquad (8)$$

where $u_Y(y)$ is utility elicited about the system capabilities.

In order for these two expected utilities to be equal for all functions, p(y | A), it must be true that

$$u_Y(y) = \sum u_X(x) \cdot p(x \mid y, A)\, dx \equiv u_{X|Y}(y \mid A). \qquad (9)$$

The integral will be referred to as the derived utility function of x given y. It is a utility function derived from the utility function, $u_X(x)$, on the mission objective variable(s), x, and the model, p(x | y, A), of achieving those objectives with a system, A, based on performance variable(s), y. This relationship is critical to understanding the relationship between mission objectives and system capabilities, and whether assessing utility on capabilities rather than objectives is valid.

4. Relating Previous Approach to MAUT

The earlier MCES mathematical approach is a special case of the MAUT approach, where the utility function is the characteristic function of a crisp set defined by the boundaries of the success region. Consider the following equations.

The measure, $\mu(R^* \mid \underline{v}, s)$, above, is equivalent to taking

$$\sum_{\underline{r} \in R^*} p(\underline{r} \mid \underline{v}, s) = \sum u_R(\underline{r}) \cdot p(\underline{r} \mid \underline{v}, s)\, dr \qquad (10)$$

where $u_R(\underline{r}) = 1$ for $\underline{r} \in R^*$, and 0 otherwise. The effectiveness measure, above, is equivalent to taking

$$E(s) = \sum u_V(\underline{v}) \cdot \mu(R^* \mid \underline{v}, s)\, p(\underline{v} \mid s)\, d\underline{v} \qquad (11)$$
$$= \sum u_V(\underline{v}) \cdot \left[\sum u_R(\underline{r}) \cdot p(\underline{r} \mid \underline{v}, s)\, d\underline{r}\right] \cdot p(\underline{v} \mid s)\, dv$$

where $u_V(\underline{v}) = 1$ for $\underline{v} \in V^*$, and 0 otherwise. Note that $u_V(\underline{v})$ depends on the shape of V*, which is an inverse mapping of the set, R*.

The functions, $u_R(\underline{r})$ and $u_V(\underline{v})$, above, are equivalent to crisp set utility functions, i.e., functions with value (worth) equal to 1 inside the set, and 0 otherwise. But the boundary between success and failure is not so cut and dried, although we have grown accustomed to defining it that way in some cases. But surely, losing 10 of 100 aircraft is not as good as losing none, and losing 11 is not total failure, as would be the interpretation of a success region of "lose no more than 10 aircraft." Rather, a smoother utility function could be defined, if

necessary. The crisp set is not precluded by the proposed approach; it is a special case. The functions, $u_R(\underline{r})$ and $u_V(\underline{v})$, could be smooth functions, as well.

Utility functions look very much like fuzzy sets or conditional probability measures, but the operations on them, in the MAUT theory, are like probability, i.e., multiplication and addition, rather than MIN and MAX as in fuzzy logic.

5. Extending The MOE Concept

The above formulation can now be extended to more general cases. Extending the mathematical formulation to other aspects of the combat domain can be enlightening in terms of relating the force effectiveness and C3 effectiveness and performance measures. In the proposed approach, performance measures will be (conditional) probability distributions, and importance is represented by utility functions, while effectiveness measures will be expected utilities.

Consider the following domain variables:

$r \in R$, the domain of results or combat outcomes,

$c \in C$, the domain of C3 system outcomes, $cs \in CS$, the set of C3 systems,

$w \in W$, the domain of weapon outcomes, $ws \in WS$, the set of weapon systems,

$s \in S$, the domain of sensor outcomes, $ss \in SS$, the set of sensor systems,

$a \in R$, the state of other assets, $as \in AS$, the set of other assets,

$b \in B$, the background state, including the environment, scenario and enemy forces.

When the choice to be made is among C3 systems for a given set of sensor, weapon and other assets, the objective function is

$$EU_R(cs) = \Sigma\, u_R(r) \cdot p(r \mid cs)\, dr. \qquad (12)$$

where the conditional probability can be determined by the chain

$$p(r \mid cs) = \Sigma \cdots \Sigma \left[p(r \mid w) \cdot p(w \mid c, ws, b) \cdot p(c \mid s, cs, b) \cdot p(s \mid ss, b) \cdot p(ws, ss, b)\right] dw\, dws\, dc\, ds\, dss\, db$$

$$= \Sigma \cdot\cdot \Sigma \left[p(r \mid w) \cdot p(w \mid c, ws, b) \cdot p(c \mid s, cs, b) \cdot p(s \mid ss, b)\right] dw\, dc\, ds, \qquad (13)$$

with other assets interspersed as necessary. The last expression assumes fixed values of the WS, SS and B variables. The chain makes sufficiency assumptions, or conditional independence assumptions, as appropriate. The equation is only notional and is not intended to declare these sufficiencies. In fact, an important change needs to be made to the last equation to provide for the control of sensors (and other assets). In order to do this, there needs to be a subdivision of the C3 outcome vector, so that $\underline{c} = \{ \underline{c}_W, \underline{c}_S, \underline{c}_a \}$. There also needs to be a feedback loop in the sensor control, so the $\underline{c}_S$ vector needs time tags, which will be indicated by + and -. Of course, causal antecedents always must have time signatures no later than the consequent variables. Suppressing the other asset elements, the last equation becomes

$$p(r \mid cs) = \Sigma \cdot\cdot \Sigma \left[p(r \mid w) \cdot p(w \mid c_W, ws, b) \cdot p(c_W, c_S^+ \mid s, cs, b) \cdot p(s \mid c_S^-, ss, b)\right] dw\, dc\, ds. \qquad (14)$$

There should also probably be a c_C term in order to handle the iterative control loop of the command system itself, such as organization changes to fit the situation, delegation of authority, etc. This will be added below.

In order to assess C3 systems, it would be nice to have an expected utility expression for the C3 systems themselves, such as

$$EU_C(cs) = \Sigma\, u_C(c) \cdot p(c \mid cs)\, dc. \qquad (15)$$

We can find

$$p(c \mid cs\,;\, ss, b) = \Sigma\, p(c_W, c_S^+ \mid s, cs, b) \cdot p(s \mid c_S^-, ss, b)\, ds, \qquad (16)$$

and,

$$u_C(c_W\,;\, ws, b) = \Sigma\Sigma\, u_R(r) \cdot p(r \mid w) \cdot p(w \mid c_W, ws, b)\, dw\, dr. \qquad (17)$$

So, it is necessary that the performance of the C3 system be conditioned on the sensor performance and the background variables, as intuition tells us. It is also a result, that the derived utility function has to be computed using weapon performance measures. This says that C3 measures cannot be expressed independently of the performance of the sensors and weapons chosen (available), and vice versa. !!!

This should not be considered as a drawback of this approach. Rather, it is a testament to its validity that this conclusion is so apparent. The worth of any system is related to the performance of systems downstream in serial with it and the context in which it is operating, while its performance is dependent on the upstream systems in series, and the context. Note that the downstream systems, weapons, influenced the utility function in equation (17), while the upsteam systems, sensors affected the C3 performance distribution in equation (16), and background affected both.

6. Definition of Measures

The mathematical elements of the Derived Utility approach provide the definition of MOEs. Let performance be represented by causal conditional probabilities relating system states, and let effectiveness

be indicated by an expected utility on objective states. Effectiveness can also be defined for derived utilities, when applied to conditional probabilities of the system performance variables. This leads to the following definitions:

6.1 MOFE:

$$EU_R(ws, cs, ss, as, b) = \sum u_R(r) \cdot p(r \mid ws, cs, ss, as, b)\, dr, \quad (18)$$

where $p(r \mid ws, cs, ss, as, b)$ is a model of performance of the force and is decomposable into submodels of system performance.

6.2 C3 MOP

Two measures are apparent, one conditioned on sensor outcomes and the other combined with the sensor performance.

Conditioned performance:

$$p(c \mid s, c_C^-, ws, cs, ss, as, b) = p(c_W, c_C^+, c_S^+, c_a \mid s, c_C^-, ws, cs, ss, as, b) \quad (19)$$

Combined performance:

$$p(c \mid c_C^-, c_S^-, ws, cs, ss, as, b) = \sum p(c \mid s, c_C^-, ws, cs, ss, as, b) \cdot p(s \mid c_S^-, ss, b)\, ds, \quad (20)$$

which is conditioned on previous decisions about sensors.

6.3 C3 MOEs

Again cs effectiveness can be conditioned on or combined with sensor performance:

$$EU_C(cs; s, ws, ss, as, b) = \sum u_C(c_W ; ws, b) \cdot p(c \mid s, c_C^-, ws, cs, ss, as, b)\, dc, \text{ or} \quad (21)$$

$$EU_C(cs; ws, ss, as, b) = \sum u_C(c_W ; ws, b) \cdot p(c \mid cs; c_C^-, c_S^-, ws, ss, as, b)\, dc. \quad (22)$$

The dependence on prior decisions has been suppressed on the left side in these equations, assuming they are specified as initial conditions. The non-C3 systems could also be suppressed for a given initial order of battle. The C3 system variable, cs, is the subject of the effectiveness measure, i.e., the alternative set, and must be retained.

Note the dependence on other systems, but the utility function seems only to depend on the c_W decisions. This needs to be examined. Similar definitions would apply to the other combat systems, attempting to isolate their causal relations and the upstream and downstream effects.

6.4 MOIs

The equating of expected utility at the top level objectives with the effectiveness of a system gave rise to the derived utility as an element of the calculation of the MOE. In [7], a measure of importance (MOI) was defined, which is the partial derivative of the effectiveness measure with respect to the performance measure of the element under consideration. The derivative of

$$EU_C(cs) = \sum u_C(c) \cdot p(c \mid cs)\, dc. \quad \text{see (15)}$$

with respect to the performance measure, $p(c \mid cs)$, is the other functional under the integral, $u_C(c)$, which is the derived utility of the command system. This leads to the definition of the MOI of a system as "the derived utility function" (within the scenario, considering the other forces and environment involved, as described above) based on the utility function of the top-level objectives.

The top-level utility function, itself, is an MOI of the overall analysis, since it is a statement of the relative importance of the outcomes of the operation. The equating of expected utility at the top level objectives with the effectiveness of a system gives rise to the derived utility as the Measure of Importance (MOI) of a system, an element of the calculation of the MOE.

The relationship of these measures is shown in Figure (2), Measures of Performance, Importance, and Effectiveness. On the left is the full hierarchy of relations of these measures. But there is a suggestion that system effectiveness (MOE(S)) is based on utility ascribed directly to system attributes, without regard for the mission effect. On the right, it shows how the derived system-level utility, computed from a Force Model, coupled with a Mission-related Utility Function, generates the same Force-level Expected Mission Utility, MOE(F), as the left side. If the Ad Hoc System Utility Function is the same as the Derived System Utility Function, then the Expected System Utility would also match the Expected Mission Utility.

7. C3 Performance Variables

What is still missing is a definition of the C3 variables. To say they deal with timeliness and other such issues, does not let us fit them into the approach as some kind of state variable. The role of C3 systems is to assist the command in making decisions. The variables that enable activity by other assets are, of course, decision states, as described in Section 4.5. A decision is a choice of assignment of an organization of people and resources to carry out a force function, for a purpose and within an allotted time. The model of decisions and,

therefor, the performance measure is a distribution of the probability of making that assignment.

The quality of the decision is not inherently good or bad. The model is a description of what occurs with various inputs. The quality of the decision depends on the quality of the inputs and the fidelity of the internal decision process to infer correctly the state of the world from those inputs and to project correctly the potential outcomes of courses of action taken. The value of the decision is dependent on the value of the actual outcome, as derived from the utility function.

8. Measures for a UAV Application

For the UAV problem, four general missions have been identified (MORS C3IEW MOE II workshop September, 1992) Target Development, Situation Development, Target (Re-)Acquisition (Targeting and Adjustment), and Battle Damage Assessment. Mission objectives gleaned from various sources are essentially the same as those of any Reconnaissance, Surveillance and Target Acquisition (RSTA) system (i.e., detect, identify, localize/track, and report targets of interest and situational and environmental conditions) in support of Air Defense, Strike, Interdiction, and Air, Ground, and Naval Fire Support missions. These objectives are refined in Air Tasking Orders (ATOs) in the case of air units such as UAVs. Ancillary objectives assayed from other writings regarding the need for UAVs include reducing lives lost (those of RECCE pilots, ones own warriors (self-attrition), and civilian or other by-standers (for moral and political reasons), reducing aircraft lost to reduce operations cost, and meeting mission schedules in order to accomplish the other objectives.

A specific set of missions was defined for a scenario being used to assess the merit of several classes of UAVs in relation to other RSTA assets and each other [8]. Target and Situation Development was reflected in missions for Land Area Sweeps out to 55 nm, Beach Reconnaissance at low tide and 3 nm inland, and Seaward Barriers at 40 nm from a central point. Targeting Support for Naval Surface Fire Support forces two hours per day was required and Battle Damage Assessment support was needed continuously by alert UAVs, able to respond in 15 minutes, aboard Naval Forces. One can see that not only were the missions more specific, some criteria such as response time are evident. Additional criteria such as percent of targets detected may be needed.

In [8], the following measures of performance were defined:

1. Number of missions flown
2. % of missions flown
3. Alert response time
4. Hours on Station
5. % of hours on station
6. Total hours of flight operations (of the support ship)

The model also kept track of other measures of performance:

1. Hours of operation (flight hours)
2. Responsive to alerts
3. Time late in arrival to missions
4. % of time continuous missions not covered

Other notable measures mentioned were:

1. Cost per target killed (weapon costs) vs. cost to find targets
2. Cost of operations
3. Cost of killing the wrong person

Other qualities of systems found in treatises on measures of effectiveness include:

Survivability
Transportability
Deployability
Counter-detectability
Vulnerability
Degree of interoperability
Reliability
Maintainability
Availability

Other conditioning variables in various studies include:

Weather (effect on sensors, flight, and launch/recovery)
Terrain (physical environments?)

System configurations to be considered as system variants, competing alternatives, or complementary support were:

Current RSTA support to lower-level units
Increased C3 connectivity to Bde/Batt
Modified SR-UAV concept of employment
Airframe (vs SR, LR?)
Payloads
Number of units produced
Number of units available to Commander
Hand-launched UAVs
Manned helicopters
Unmanned ground vehicles

9. Working with the Structure of Measures

The problem being posed is how to put this collection of missions, objectives, qualities, and performance measures into a consistent, logical, relatable structure. The conditional utility approach would ask, for example, what is the effect on expected operational outcome utility of having a higher percentage of on-station time? This does not imply that percentage of on-station time is unimportant; it attempts to place that measure in perspective with respect to the purpose of the system, which is not to be on station, but to collect important

information. On-station time may be the only discriminant between competing UAV systems, assuming payload performance is identical, but the value of the system may need to be compared to another type of system. In that case, the measure of merit needs to be based on objectives above the level of the systems being compared.

9.1 A Simple Example

Figures (3) and (4) show direct and derived utility in terms of an acoustic surveillance system. The approach requires defining the performance and qualities of the system in terms of conditioning state relations. For example, Availability is the state of being available. A system is available if it is in a state of "good repair" and "not busy", the latter depending on the number of units in theater and the number of missions demanding service. Maintainability is a measure of the probability of being in "good repair", given a previous failure, while Reliability is a measure of the probability of failure, given a previous "good repair" state, also conditioned on the operation being performed and the environment. Measures such as these tend to depend on a factor of the time since the last state change. With these definitions in hand, the system performance can be characterized as depending (conditioned) on Availability and the probability of Availability can be factored into the equation,

$$\begin{aligned} &P(\text{Success} \mid \text{Other variables}) \qquad (23)\\ &\quad = P(\text{Success} \mid \text{Avail,...other variables}) \cdot P(\text{Avail})\\ &\quad + P(\text{Success} \mid \text{Not Avail,...other variables}) \cdot\\ &\qquad\qquad P(\text{Not Avail})\\ &= P(S \mid A, o) \cdot P(A) + P(S \mid NA, o)\,[1 - P(A)]\\ &= [P(S \mid A, o) - P(S \mid NA, o)] \cdot P(A) + P(S \mid NA, o), \end{aligned}$$

while

$$\begin{aligned} &P(\text{Avail } (t_1)) = P(\text{good repair } (t_1) \mid \text{failure } (t_f))\\ &\quad + 1 - P(\text{failure } (t_1) \mid \text{good repair } (t_g)) \qquad (24) \end{aligned}$$

is a model of the state transition between repair and failure, conditioned on environment and employment (not expressed explicitly). The *sensitivity* of the success measure to the availability measure is the derivative of equation (23) with respect to P(A), which is the difference in performance for the cases that the system is available or not:

$$P(S \mid A, o) - P(S \mid NA, o).$$

9.2 A Higher-Level View

At the higher levels, the top level performance measure is the probability of accomplishing the mission's purpose (to some degree), while the effectiveness measure is the expected utility of the accomplishment over all degrees. The latter is the probability of outcomes times the satisfaction of each outcome, summed over all outcomes. Detecting 90% of targets has a higher degree of satisfaction than detecting 50%. How much satisfaction is achieved depends on what can be done with that information. One's value of the information should depend on one's model of how the information will change the outcome at a higher level, such as number of targets killed, given 90% detected. The number of targets killed has a varying level of satisfaction, also, perhaps even leveling off at some point. Since there are objectives such as reducing lives lost, these too have value functions which are needed to weight the probability of losing lives. The shapes of these individual worth functions will effect the resulting expected utility.

For example, for simple success/failure assessment, the effectiveness measure is

$$\begin{aligned} &EU(\text{System A}) \qquad (24)\\ &\quad = V(\text{Success}) \cdot P(\text{Success} \mid \text{System A}) +\\ &\qquad V(\text{Failure}) \cdot P(\text{Failure} \mid \text{System A})\\ &\quad = [V(\text{Success}) - V(\text{Failure})] \cdot\\ &\qquad P(\text{Success} \mid \text{System A}) - V(\text{Failure}) \end{aligned}$$

where V(Success) and V(Failure) are the two values of the utility function. The importance (sensitivity) of this measure to success is the derivative with respect to P(Success | System A). This is the difference in the value function elements:

$$V(\text{Success}) - V(\text{Failure})$$

This says that if we are indifferent to the outcome (difference = 0) the importance of the system is nil, but it increases as the value of success is much greater than the value of failure.

We can also examine the effects of systems in series and parallel configurations. If the success depends on the *series* performance of two systems:

$$\begin{aligned} &EU(\text{System A, System X}) \qquad (25)\\ &\quad = \{[V(S) - V(F)] \cdot P(S \mid A) \cdot P(S \mid X)\} - V(F), \end{aligned}$$

where S and F have replaced "Success" and "Failure, respectively." The importance of each system is now equal to

$$[V(S) - V(F)] \cdot P(S \mid \text{other system}).$$

While there is still dependence on the relative value of the outcomes, it is now apparent that the importance of a system is greater if the other system's performance is good than if it were bad. The focus of attention should be on the system with the worse performance of the two.

In contrast, for systems which contribute in *parallel* to success, attention should be paid to improving the better system, unless the marginal cost of improvement is prohibitive. (This leads to the need to address a sensitivity measure which includes the marginal cost of improvement. This measure would involve the product of the derivative of the effectiveness measure with

respect to system performance and the derivative of the system performance measure with respect to cost, as well as the utility function on cost.)

9.3 A Higher-Level Example

The example presented in Section (9.1) had serial and parallel components. The performance of the mission was recognized as

$$P(S^* \mid \text{Sensor, Comm, C2, H1, H2}) = P(S \mid \text{Sensor}) \cdot P(S \mid \text{Comm}) \cdot P(S \mid C2) \cdot [P(S \mid H1) + P(S \mid H2) - P(S \mid H1) \cdot P(S \mid H2)] \quad (26)$$

where S* represents mission success and S means system success. The effectiveness of the mission is

$$EU(\text{Sensor, Comm, C2, H1, H2}) = \{V(S^*) - V(F^*)\} \cdot P(S^* \mid \textit{systems}) - V(F^*) \quad (27)$$

where the importance of the set of systems is the difference in the value of the outcomes of Success and Failure: $\{V(S^*) - V(F^*)\}$.

The importance of any of the C3I functions (sensor, comm or C2) is the product of the difference in the value of the outcome, the performance of the other two C3I systems and the compound performance of the helicopters. For example, the importance of the C2 system is

$$\{V(S^*) - V(F^*)\} \cdot P(S \mid \text{Sensor}) \cdot P(S \mid \text{Comm}) \cdot [P(S \mid H1) + P(S \mid H2) - P(S \mid H1) \cdot P(S \mid H2)]$$

The importance of each component of the C3I chain depends on the performance of the other components. The importance of each helicopter is the product of the value difference, the performance of the C3I chain and (1 - P(S | H(other))), so, the better the performance of the other helicopter the less important is the one under consideration. For the helicopter(1), its importance is

$$[V(S^*) - V(F^*)] \cdot P(S \mid \text{Sensor}) \cdot P(S \mid \text{Comm}) \cdot P(S \mid C2) \cdot [1 - P(S \mid H2)]$$

Comparing the two helicopters (in this case) does not depend on the other systems or the value function, since they are considered to be the same for both cases. This only applies for comparing systems with the same function. To compare different system types requires focusing on the distinct elements in their importance measures, in this case:

$$P(S \mid C2) \cdot [1 - P(S \mid H2)],$$

for the helicopter, and

$$[P(S \mid H1) + P(S \mid H2) - P(S \mid H1) \cdot P(S \mid H2)]$$

for the C2 system.

10. Conclusion

The power of the approach has been demonstrated, through this example, to allow focusing on different aspects of an analysis using the same few principles of conditional modeling, measure definition and sensitivity assessment. It demonstrates the intuitively correct notions of the value of series and parallel functions and the dependence of a system's contribution to mission success on the performance of other systems. A modeling tool that embodies these powers would be a valuable investment.

The proposed approach to measures is the first to provide a morphological basis for distinguishing between MOPs and MOEs, in that the former is a distribution and the latter is an expected utility. The chain of conditional relations is an effective way to perceive the functional structure of systems and objectives. It provides a perspective on measures also, in that it makes clear the dependence of one measure on another and whether one is subordinate to another or if there is redundancy in measures. The definition of information states for decision systems provides a way of coupling decision models into the analysis. The newly derived measure of importance (MOI) provides a tie between MOPs and MOEs that is an innovative and essential element of analysis.

The derivation of the utility of the decision in the context of the mission objectives satisfies a goal of the author to understand this relationship. This goal was motivated originally by a paper by Signore and Starr [9] which focusses on the mission as the purpose of the system.

11. References

[1] Girard, P., "Command and Control Systems Requirements Analysis: The Hierarchy of Objectives Approach", Naval Ocean Systems Center Technical Document (TD 1938 Vol 1), September 1990

[2] Girard, P., "Command and Control Systems Requirements Analysis: Measuring C2 Effectiveness with Decision Probability", Naval Ocean Systems Center Technical Document (TD 1938 Vol 2), September 1990

[3] Girard, P., "Command and Control Systems Requirements Analysis: Command and Control System Functions in the Hierarchy of Objectives", Naval Ocean Systems Center Technical Document (TD 1938 Vol 3), September 1990

[4] Sweet, R., Sovereign, M., and Metersky, M., *Command and Control Evaluation Workshop*, Military Operations Research Society Report, (Revised) June 1986

[5] Girard, P., "Deriving Utility of (C2) System Capabilities from Mission Success Criteria", presented at 62nd Military Operations Research Society Symposium, Colorado Springs, Colorado, June 7-9, 1994

[6] Elele, Dr. J., Dr. M. Sovereign, Dr. P. Girard, "Measure Based Modeling", Technical Report based on MORS Workshop on C3IEW Measures, Monterey, California, 1992.

[7] Elele. J.. "Methodology for Quantifying the Contributions of Command. Control. Communications. and Intelligence Systems and Subsystems to Overall Mission Performance" (with J. Cole). *Proceedings of the U.S.Army Operations Research Conference XXX2.* Fort Lee. VA. Vol. 1B. pp. 31-44. 1993.

[8] Sovereign. M.. and Nakagawa. Preliminary Outline and Notes on UAV Options. September 1992

[9] Signore. David T.. and Stuart Starr. "The Mission Oriented Approach to NATO C2 Planning". *Signal, vol. 42 no.1*.September 1987

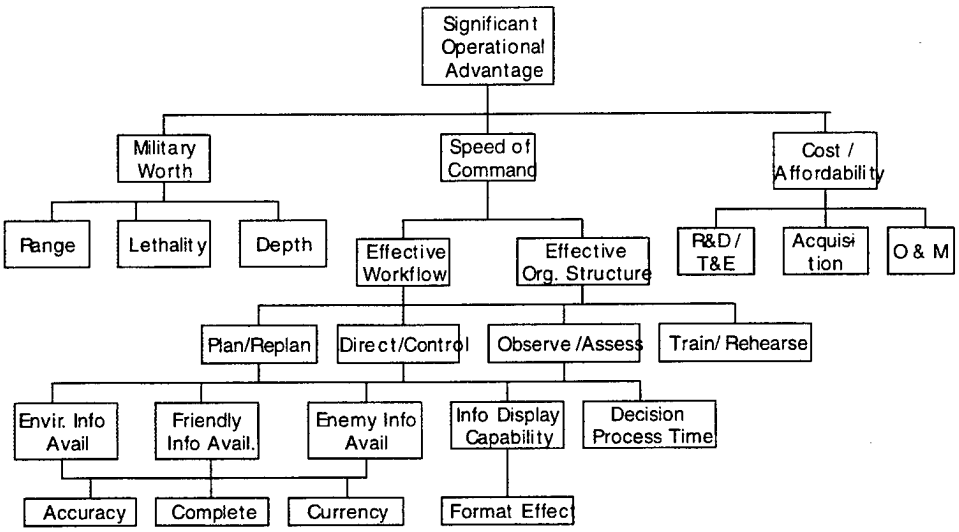


Figure (1) JFAC Mission Hierarchy of Purpose

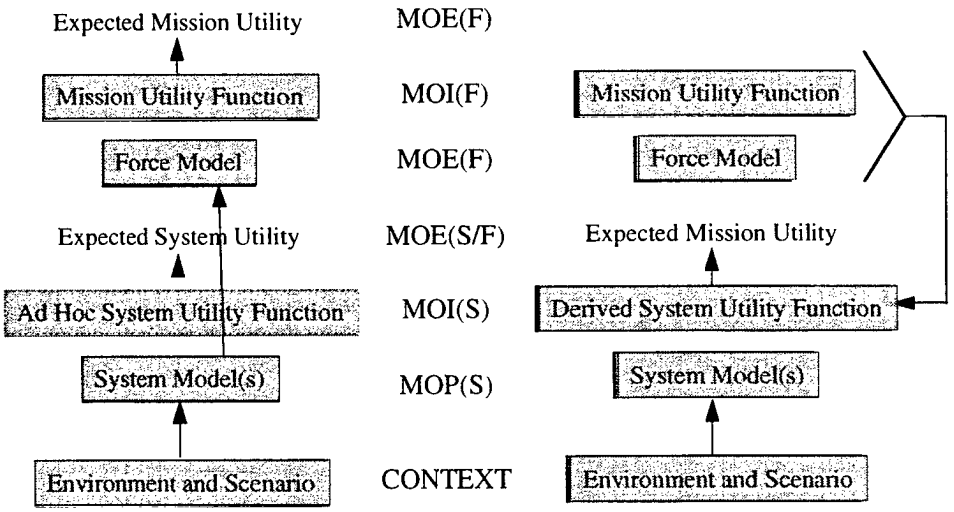


Figure (2) JFAC Mission Hierarchy of Purpose

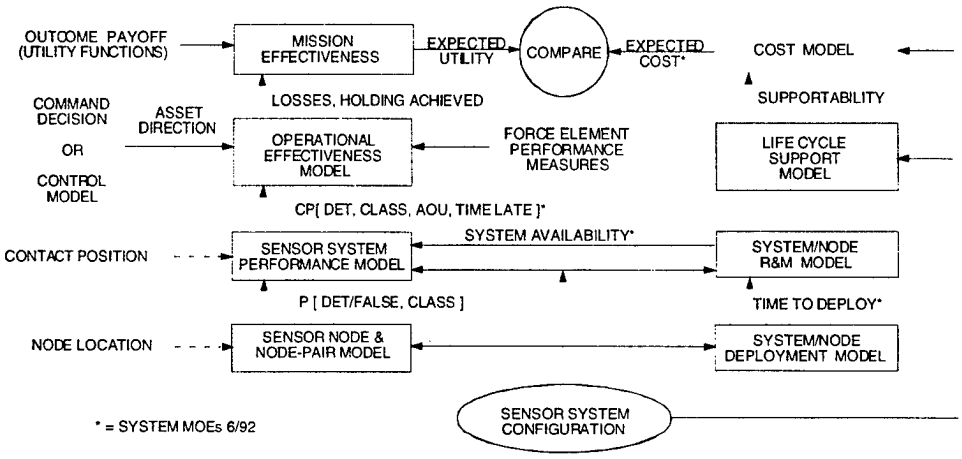


Figure (3) Direct top-level utility approach

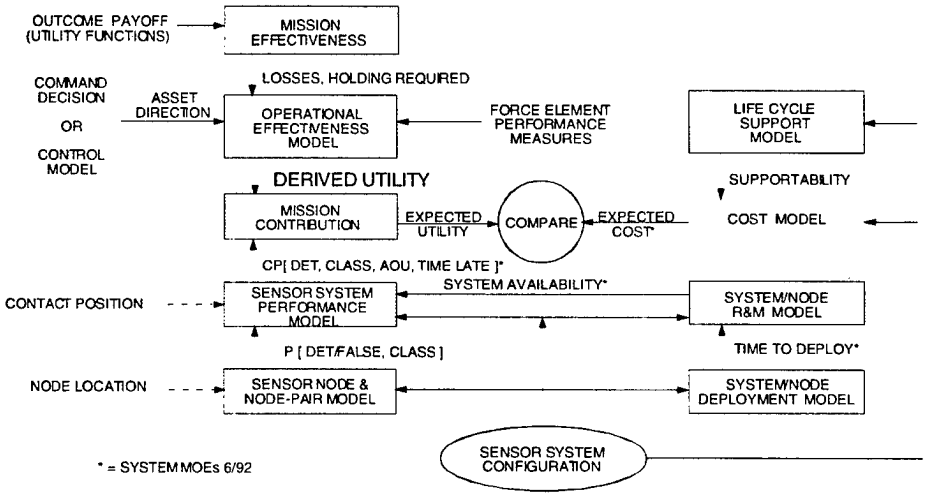


Figure (4) Derived Utility approach

A Consolidated Decision Process Paradigm for Modeling C3 Systems

Dr. Paul E. Girard
Science Applications International Corporation
4015 Hancock Street, M/S T-1
San Diego, CA, 92110
(858) 826-7158, FAX: (858) 826-5617
paul.e.girard@saic.com

Abstract

The paper presents a paradigm for conceptualizing the C3 decision process. The functions of Plan, Observe, Assess, and Execute (POA&E) consolidate many features of earlier concept models. It provides a second level of detail to reveal the inner workings of the top level, and it indicates the role of modeling, mental or otherwise, in the decision process. This paradigm will be useful in organizing thinking about the architecture of the Enterprise Model, in terms of collaboration and information flow among decision elements performing each (and every) POA&E function. The conceptual model can be applied to aggregate units (i.e., command centers) or to individual functions within those units, or to individuals and specific decisions.

1. Introduction

A conceptual model of the decision process, called the Command Process Model (CPM), was described in [1] through [4] and is summarized in the following paragraphs. This summary provides an overview of the relationship of the functions among themselves and to the operations they control. These functions were derived from other models of the decision process. The CPM consolidates important features of those concepts. A discussion of those features is provided, to highlight those aspects of the Command Functions and to explain the role of those features in the structure of the decision process.

2. Command Process Model

The CPM consist of four major functions: Plan, Observe, Assess, and Execute, sometimes abbreviated as POA&E. These one-word function names are used for ease of discussion. Additional functions of Sense and Act may be appended to the list to couple the other four to the physical world. The alternative functions of Receive and Issue (or Send) play a similar role in coupling to other decision-making activity.

The CPM is depicted in Figure (1), Command Process Model. The decision process, the heart of the CPM, is in the center, with the coupling functions around it. The left side collectively performs monitoring activities, while the right side constitutes the control portion. The main sequences of the decision process are shown by arrows. Information flows among the decision functions are not shown, but they are described in the references.

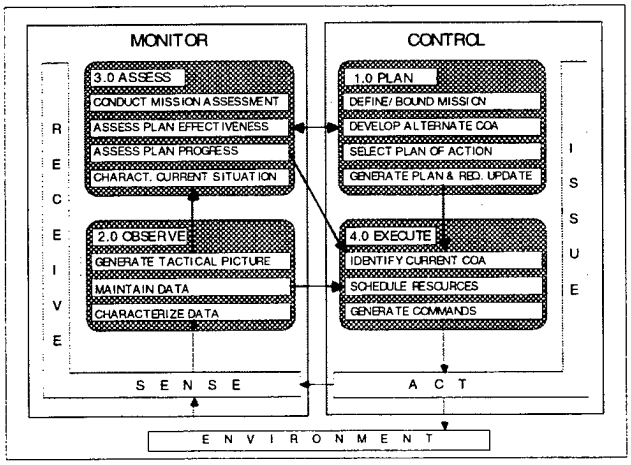


Figure (1) Command Process Model

Information obtained through the Receive function can be destined for any of the decision functions. Information can also be obtained from the Sense function as an input to Observe. Information is transferred through the Issue function. Actions being controlled are defined in the Act function. This includes control of the sensing function, which produces sensory inputs. It is not critical, whether Sense/Receive is viewed as a single function or as two functions. Likewise for Issue or Act. But there is a subtle difference, so they are shown separately.

2.1. Observe

The Observe function combines information for use by other functions. This is an all-encompassing "data fusion" function. It involves not only storing data together, but also association, correlation, and tracking functions and compilations of Intelligence data and Force capability and

status information. This aggregate of information is often referred to as the Tactical Picture at the combat level.

2.2. Assess

The Assess function makes use of the combined data to infer meaning about the situation, including enemy intent and potential outcomes of unfolding events. This is the real product of Situation Assessment, not just the Tactical Picture produced by Observe. The assessment determines whether mission objectives are being achieved, a new or revised set of plans is required or a change of procedure under current plans is appropriate. The overall Mission Assessment might result in notification to higher authority that the Mission objectives are unachievable, or a change in objectives or constraints is necessary, or additional resources are required. If planning is required, the Plan function is invoked. If a change in procedure is suggested by the situation, the Execute function is notified and can change modes, possibly in accordance with a contingency plan. Otherwise, Execution proceeds under the current mode, using information from Observe.

2.3. Plan

The Plan function generates optional courses of action intended to achieve the mission. Based on the same kind of assessment that the Assess function produces about projected mission achievement, the Plan function evaluates and selects primary and contingency courses of action, including organizational responsibility, procedures and allocation of resources to general task areas or specific tasks. The criteria for assessing situations and changing procedures are defined by Plan for use in Assess in determining when these conditions exist. The procedures, including rules for allocating resources, are used by Execute to implement the plan and control its progress.

2.4 Execute

The Execute function selects a specific course of action, based on the current assessment of the situation (from Assess). Using procedures established in the plan and data from Observe, specific allocation of resources and tasks or even specific guidance variables are generated as directives and Issued as orders or implemented as Actions. It is some form of a resource allocation and activity directive (an allocation of resources to perform tasks in pursuit of objectives) that represents the product (outcome) of the decision-making process.

3. Features of the CPM

The principal feature to note about the CPM structure is the set of three paths through the decision process. These represent three fundamentally different kinds of decision/control cycles. The outside or largest loop consists of passing through all four functions, while the next loop bypasses the Plan function and the inner loop consists of only parts of the Observe and Execute functions. These loops can be called planning control, mode control, and base control, respectively. The planning loop involves the generation and consideration of multiple courses of action within the Plan function and the selection of a subset of these as a set of contingency plans that can be considered for implementation in the Execution function. Each contingency or branch of a contingency can be considered as a mode of operations that can be activated depending on the situation. The changing of modes is dependent on the assessment of the situation in the Assess function. This dependency is represented by the path from the Assess function to the Execute function in the mode control loop. (This loop is the most important difference from the previous Framework document functions, as will be discussed below.) For a given mode of operation, the base control loop generates directives that are appropriate for that mode, using information from the Observe function.

The separation of the planning control and mode control loops is an important concept in effective decision making. Both Plan and Execute have a role in selecting courses of action. The planning loop involves generation, consideration, and selection of options, but the final decision is deferred until execution. Execute makes that decision, as well as the sequential decisions needed to carry out the selected option. The planning loop is a deliberative one and takes time. Most decisions are made without going through such a process, but rather by choosing among courses of action that are predetermined by prior planning or that are familiar or preferred, due to previous experience (and success). At the same time, the planning loop provides for dealing with the unanticipated situation.

4. Relation to Other Decision Models

The functions of the CPM were derived as an amalgamation of several models of decision making. An Army source [5], uses Acquire-Assess-Determine-Direct. An Air Force perspective [6] cites the Observe-Orient-Decide-Act loop of Colonel John Boyd. These are simple. single-loop models without internal feedback or feed-forward. Their functions correspond roughly to those of the CPM, but decisions are represented only at one level. More complex models include the Stimulus-Hypothesis-Option-Response (SHOR) model attributed to the late J.

Wohl [7], the Lawson model [8] of Sense, Process, Compare, Decide and Act and the Headquarters Effectiveness Assessment Tool (HEAT) [9] of Monitor, Understand, Plan, Decide and Direct. Each of these emphasizes a different aspect of the process, which the CPM represents. Each has an implicit or explicit Input-Output, Sense-Act or Stimulus-Response functionality to couple the model to the rest of the world. The inputs are data of one form or another, including directives. The outputs can only be data or directives, also, so they can only cause action or possibly other decisions, but they are not the physical action.

4.1 Relation to Wohl's SHOR Model

The SHOR model (figure (2)) emphasizes the Hypothesis-Option decisions as a Yin-Yang of the process. Both involve generation, evaluation and selection (decision) of alternatives; of what to believe or what to do, respectively. These are seen as the processes of dealing with uncertainty of information and uncertainty in outcome. These can also be viewed as interpreting the past (what has been) and predicting the future (what will be), except that option selection initiates a response to "cause" the future and the hypothesis evaluation may involve projections of the current situation.

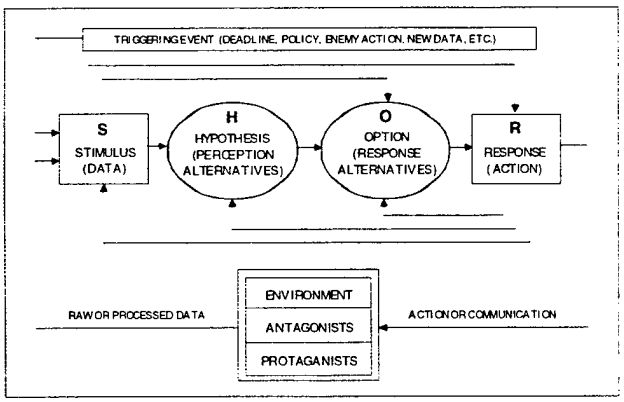


Figure (2) Wohl's SHOR Model

The CPM recognizes these two halves of the model, putting Observe and Assess under the Hypothesis or Monitor domain and Plan and Execute under the Option or Control side. The reason for the differences is to indicate that there are layers in the decision process.

The Observe function focuses on the data fusion problem of combining "sensory" inputs to form a "perception" of the location and activity of objects in the environment. (The SHOR model put much of the data fusion process in the Stimulus function, but there are hypotheses involved in data fusion, too.) This perception may be colored by expectations arising from knowledge of current plans and activities or of past behavior of the subject elements. It may even be biased by current assessments. The Assess function infers meaning from the observation, including intent. This is a situation assessment and hypotheses involve either current situations or, possibly, they anticipate future outcomes that may occur if events unfold in ways projectable from the current situation.

On the Option side, the Plan function contains the option generation and evaluation processes. The selection of options is a decision that can possibly result in contingency plans that become deferred decisions or preplanned courses of action. The final choice is made in the Execute function, after events have unfolded and/or more information is available to make a better prediction of the outcome. (The Execute function involves the choice among previously conceived options in order to reflect an aspect of the HEAT model to be discussed below.) The assessment of options requires the same type of process as the situation assessment that projects outcomes as described above. The only difference is that in one case the projection is made with respect to the currently active option, whereas a planning assessment or even a contingency assessment involves the active option and alternative options recently or previously devised. Plan and Execute also have subfunctions to generate the plans and directives that are the products of the process. In the SHOR model, these are both part of the Response function.

4.2 Relation to the Lawson Model

The emphasis of the Lawson model (figure (3a)) is on the comparison of the current (or predicted) state to the Desired State.

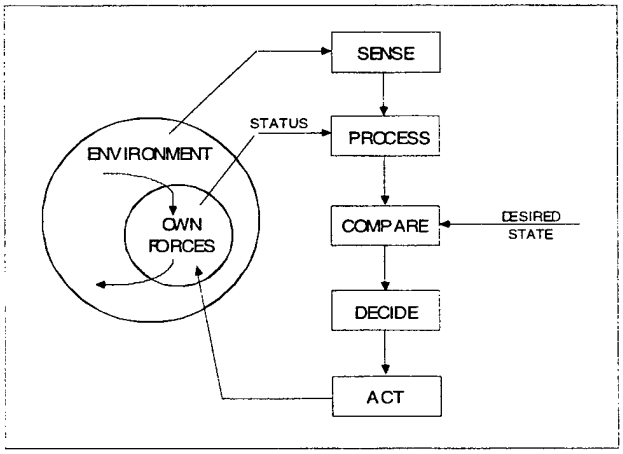


Figure (3a) Lawson's Decision Process Model

The model is basically an analogy of a control system. The Sense function takes measurements of the plant (situation) for the control system, while Process involves deriving the state of the Plant model using the measurements to determine a current state estimate. When there is no difference between the current state and the desired state, no change in control is needed. However, when there is a difference, the controller "Decides" what control signal to send (Act) to the plant to cause it to converge on the desired state.

Figure (3b), shows a Modified Lawson model based on the CPM functions. The Process function is replaced by Observe. The concept of Compare is implicit in the Assess function. The Desired State comes from some "higher level" in the Lawson model, but the Plan function generates it in the CPM based on plans and directives from the higher levels. The active option is the basis for comparison for current control activity. The Plan function also lays out how to execute the control. The Plan function, in addition, generates several possible states for comparison, any of which can be the basis for control changes, depending on the situation. Then the comparison is done relative to the various desired states and may cause a change in control mode within Execute, which replaces Decide.

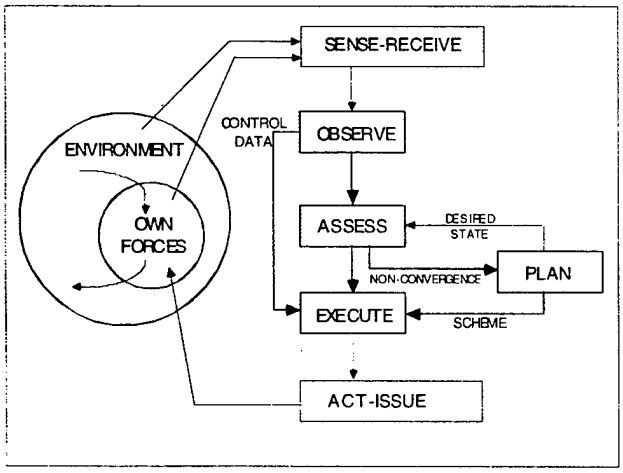


Figure (3b) Modified Lawson Model

4.3 Relation to the HEAT Model

The HEAT model (figure (4a)) emphasizes the role of planning in the decision process. The original model had all planning processes, including option generation, prediction and decision, as part of every decision cycle. In addition, lower level direction activity occurred in a simple stimulus-response loop. The measure of headquarters effectiveness was initially considered to be the lifetime of a plan, since the plan was the product of the headquarters process. Lifetime was measured to the point when the plan had to be changed due to unforeseen circumstances or ineffective courses-of-action, causing an incongruence with achieving the objective. Later, it was realized that there was a degree of incongruence that was adjustable within reason and should not be classified as ending the life of the plan. A path from MONITOR to PREDICT was inserted to reflect this idea. This results in three classes of control loops in operation in the decision process, a small loop between stimulus (sense) and response (act), a large loop which performed planning and replanning when major deficiencies (incongruences) were recognized in the plan, and a middle loop which made adjustments to the plan when minor deficits (incongruences) were noted. These are related to three kinds of decision behavior including reflexive, trained and cognitive aspects.

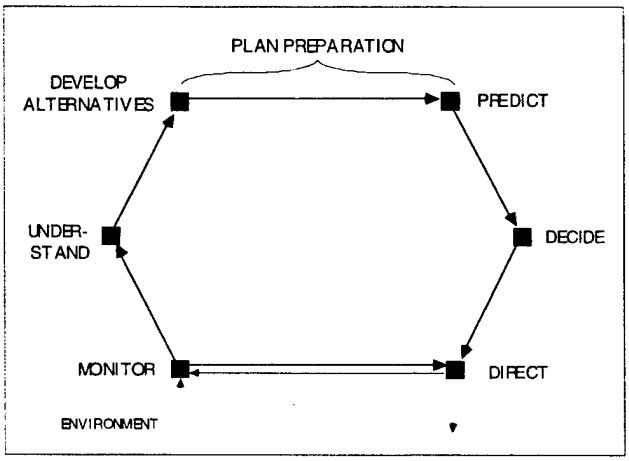


Figure (4a) HEAT Model

The present CPM paradigm includes a reflexive loop between the sensor and the effector, a trained path (the Observe -> Execute loop), and two kinds of cognitive paths; one which understands changing modes or contigencies of previously planned responses (Assess -> Execute) and the other which recognizes the need to formulate and program (plan) a new mode of operation (Assess -> Plan). This would suggest that the middle loop in the HEAT paradigm should run from the UNDERSTAND to the DECIDE functions. This is shown in a modified HEAT paradigm in Figure (4b).

The concept of prediction did not appear explicitly in the CPM, although it was expressed as an element of assessment in the verbal description. But the HEAT paradigm did not tie the PREDICT function into UNDERSTAND (taken as ASSESS). There was also no concept of contingency planning shown, although the idea of minor incongruencies suggests that with good planning, such that, if situations had been anticipated and preplanned contingency options generated, predetermined

decisions could be implemented more quickly and with less confusion. To put the PREDICT function in line with ASSESS and provide for contingency selection in the planning process, an extended HEAT paradigm is presented in Figure (4c).

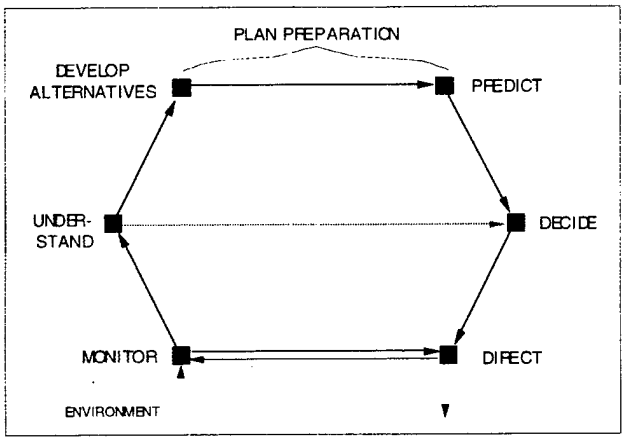


Figure (4b) Modified HEAT Model

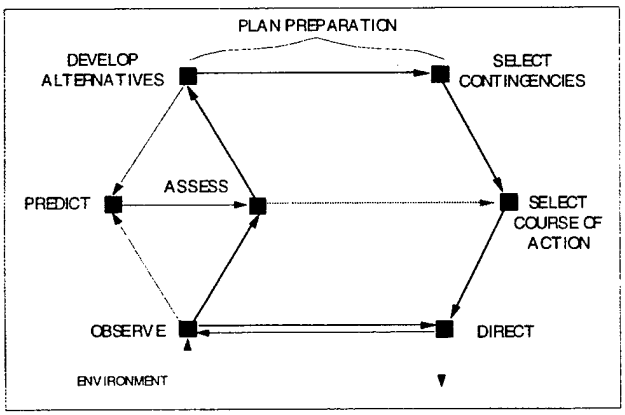


Figure (4c) Extended HEAT Model

This figure puts PREDICT (Model) into a role which supports planning by projecting the effects of alternatives, and it also serves to support current situation assessment, in as much as it treats the current plan as the operative set of alternative courses-of-action. This figure also puts the selection of contingencies (organized sets of COAs, with criteria for selecting among them, when situations or schedules dictate) as a follow-on to the generation and assessment alternatives and would depend on the measure of goodness of the outcomes projected by the PREDICT function.

5. Implications of the CPM

The Command Process Model is intended to provide an organized set of functions that can be used as a structure to define internal and external outcomes of the decision process. It is also intended to provide insight into important considerations for understanding of the decision process. This, too, will aid in the definition of outcomes.

Several kinds of decisions are involved in the decision process. Some of these will be the kind of decision that causes physical state changes or emissions, through Act and Issue. Some of them will cause changes in other information states, i.e., other decisions. These occur throughout the Command Process Model.

5.1 Two Kinds of Decision Making

There are two kinds of decisions that are made within the decision process: intentional and inferential. The main purpose of decision making is to initiate a function that is intended to cause a change of state or an emission, which, in turn, is intended to have further effects, internally or externally. This is an "intentional" decision. But, in reaching those decisions, it is often necessary to infer, from available information, the nature of the stimuli or states of the system and its surroundings, including other systems. These are "inferential" decisions. In general, the difference is that the former is a "decision to do something" and the latter is a "decision that something is true or false" or a "decision to believe something". Intentional decisions involve resource allocation and direction; while inferential decisions concern data interpretation and situation assessment. Inferential decisions are made in Observe and Assess, while intentional decisions are made in Plan and Execute. These two kinds of decisions are synonymous with hypothesis selection and option selection. These represent the two sides of the model: the inferential side which monitors and the intentional side which controls.

In the base control loop, inferences are made in Observe relative to the tactical picture or force readiness, while Execute makes intentional decisions regarding resource-task allocations and direction. In the mode control loop, Assess makes inferential decisions about the situation to which Execute responds with an intent to change operations. In the planning loop, these two kinds of decisions are performed at a higher level. The Assess function infers that current trends are not converging and Plan generates a new course of action intended to correct the problem.

5.2 Information Flow and Control Flow

The two kinds of decisions correspond, roughly, with two kinds of data flow: Information Flow and Control Flow. Information is essentially what can be inferred from data. Control is a form of data that results from intentional decisions. Since inferences are made in Observe and Assess, two kinds of information can be identified as simply observations and assessments. On the control side, there are plans and directives. The differences are subtle and only important in a relative way. Organizational relationships determine how information and control are used in a system.

5.3 Authority/Responsibility

It is well known that authority can be delegated but responsibility can not. A Commander is responsible for the actions of subordinates, so he can delegate the authority to make the decisions to perform them. While the lack of authority does not prevent action, it inhibits it in a well-disciplined organization, while having authority enables decision-making action. It is important to have the right amount of authority delegated to the right level at the appropriate time. With respect to the Command Functions, responsibility and authority are characteristics of control, involving Plan and Execute. In particular, authority is a condition for making intentional decisions, though not always required.

5.4 Coordination and Synchronization

Coordination among organizational elements is not a separate function, but involves performing the decision process functions as a joint effort. In effect, coordination is a multi-element planning, decision or synchronization activity. It is accomplished by performing the decision process individually, sharing results, identifying conflicts or mutual support requirements, and revising plans to make them consistent. Synchronization is done by using times, events or signals to initiate action. Each of these steps is individually contained in the Command Functions.

5.5 Cost of Change

An important factor must be taken into consideration when making a change in control mode or changing plans. That is the cost of change [10], which is the time to implement the change and the possibility that the time or the change itself will cause additional losses due to delay or confusion. This factor must be considered in the assessment of the situation and the replanning or mode control cycles. Another important point is that this factor is mitigated by good contingency planning and highlights the value of such planning. Having prepared for contingencies reduces the time needed to implement them and reduces the chance of confusion, if they are simply stated and familiar to the organizational elements.

5.6 Role of Modeling in the CPM

The Extended HEAT paradigm suggests that the CPM needs to be augmented with a Modeling function which supports Assess for both current situation assessment, as well as considering alternative COAs during Planning. This extension is shown in Figure (5).

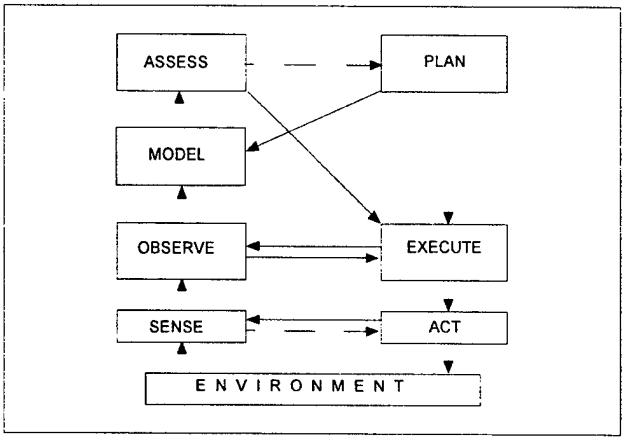


Figure (5). Extended CPM with Modeling.

6. Modeling States of a C3 System

A C3 system, like any other system must be described by its states and its processes. The CPM description recognized that there were two kinds of decision making, inference and intention. This suggests that there are two sets of state variables in a decision process, inference states and intention states. Each of these domains of states are necessarily an identical space to the space of actual states of the world (e.g., number of entities, their location, readiness to surrender, etc.). The inference space is augmented with a measure of uncertainty, while the intention states have a measure of desirability.

A third set of states for C3 systems consists of the (actual) states of the message traffic in the communications system. (Note that the physical states of the C2 and Communications systems are part of the actual state of the world.) The message states are replicas of the states of the decision process, i.e., it consists of copies of the inference and intention states as they are disseminated to others. Thus, there are three sets of states that represent the C3 portion of the functionality: Monitor (Inference) States, Control (Intention) States, and Communications States. The processes are the CPM

functions described above and the processes of the Communications system.

The Monitor States represent the inference side of the decision process. They consist of a subset of the "actual states", but could include all of them. These are belief states; that is, they should be read as "belief that the state of the world is.....". A subset of the states of the world can be observed directly. Sensor models consist of states of measurability quantities which are the primary "observable" states of information, such as bearing and range of contacts. The data fusion models represent a set of conditional probabilities about what can be inferred from the primary states of information, i.e., number of contacts believed to exist, their position and velocity, and their identity. Situation assessment is modeled as states concerning enemy intent and the potential for success, given the current plan of action.

The Control States represent the intention side of the decision process. They also represent a subset of the "actual states", but they cannot consist of all of them because they are limited to those that are "controllable". However, the "intent to cause the state of the world to be...." is an interpretation that could apply to the intention decision set of all "actual states". The control models include a resource allocation probability matrix, which could be a deterministic batch process, such as a Greedy algorithm, or it could be an event driven model which only allocated resources as they became available and tasks became "known". The model might also divert assets that are not currently available, but performing lower priority tasks. The time of the recognition that a situation exists which calls for resource allocation and the time to perform the allocation should be modeled.

The Communications States consist of the information and control states contained in messages. The key effects that need to be modeled are the probability of delivery and the distribution of delay times for a given configuration and other loading.

The Monitor, Control, and Communications States are, of course, also "states of the world", but they have a special nature in that they represent "information states", rather than physical states or abstract "mission states" that characterize the world external to the C3 elements. The sensor, weapon, and communications states and models represent an interface between the "ground truth" world and the "information" world. The information world consists of the inference information, the control information and the messages that propagate the first two.

Models of a C3 system, in keeping with the concept of stochastic conditional objects [11] (a method of modeling by propagating distributions rather than Monte Carlo realizations or expected values), involve identifying the conditional relationships between the input, internal, and output states of the C3 system. This can be stated as the probability of choosing a particular course of action and issuing a control directive with particular values, given the information input distribution and the internal information and control states. There may also be conditional models of the internal processes of the decision process.

7. References

[1] Girard. P., Command. Control. Communications and Intelligence Operational Requirements Framework: Command Process Model. Naval Ocean Systems Center Technical Document (TD 1309), July 1988

[2] Girard. P., Command and Control Systems Requirements Analysis: The Hierarchy of Objectives Approach. Naval Ocean Systems Center Technical Document (TD 1938 Vol 1). September 1990

[3] Girard. P., Command and Control Systems Requirements Analysis: Measuring C2 Effectiveness with Decision Probability. Naval Ocean Systems Center Technical Document (TD 1938 Vol 2). September 1990

[4] Girard. P., Command and Control Systems Requirements Analysis: Command and Control System Functions in the Hierarchy of Objectives. Naval Ocean Systems Center Technical Document (TD 1938 Vol 3). September 1990

[5] Dacunto. Col. L. J., "U.S. Army C2 System Master Plan and Status", in *Proceedings of 5th Annual Workshop on C2 Decision Aiding*. Army Research Institute Field Unit. Fort Leavenworth. September 1988

[6] Orr. Maj. G. E., Combat Operations C3I: Fundamentals and Interactions. Air University Press. July 1983

[7] Wohl. Joseph G.,"Force Management Decision Requirements for Air Force Tactical Command and Control" *IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-11* . September 1981.pp. 618-639

[8] Lawson. Dr. Joel S.,Jr., "The Art and Science of Military Decision Making", *Phalanx, vol. 15, no. 4*. December 1982

[9] Headquarters Effectiveness Assessment Tool (HEAT) Description and Briefing Materials. Defense Systems. Inc., No date.

[10] Snyder. Dr. Frank. Naval War College. Personal Communication. November 1988

[11] Girard. P., "Stochastic Conditional Object Oriented System Analysis". Proceedings of the Joint Directors of Laboratories C2 Research Symposium. June 12-14, 1990.

On the Role of Humans in Enterprise Control Systems: the Experience of INSPECT

Andre Valente, Jim Blythe, Yolanda Gil and William Swartout
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
{valente,blythe,gil,swartout}@isi.edu

Abstract

In this paper, we use the example of a successful mixed-initiative plan evaluation tool for the domain of air campaign planning to argue that the human-in-the-loop is an important feature of enterprise control systems. Our tool, called INSPECT, *evaluates air campaign plans and alerts the user about inconsistencies and potential problems. A generalization of* INSPECT *called PSMTool is also capable of limited interaction with a subject matter expert to capture new critiques of plans. The paper describes our work on* INSPECT *and PSMTool, analyzes the key contributions of these tools, and draws some conclusions about the role of mixed-initiative tools in enterprise control systems.*

1 Introduction

Working with Air Force experts from the "Checkmate" Group at the Pentagon (HQ USAF XOOC), we developed INSPECT, a tool for evaluating and critiquing the air-related portion of a military campaign plan. INSPECT integrates several AI technologies. It was built using the EXPECT framework for knowledge-based systems development, that incorporates knowledge acquisition techniques, a description logic-based knowledge representation system, and a sophisticated problem-solving language and reasoner.

Our experience in developing INSPECT and several other systems led us to argue for *mixed-initiative systems* as the right approach for building enterprise control systems. The air campaign planning domain experts we worked with felt that it was essential that any system should keep human users "in the loop." For example, they argued that tools for air campaign plan evaluation should allow users to understand their plans, identify critical factors, and analyze tradeoffs among options. More importantly, these domain experts argued that evaluation criteria need to be adapted in response to new situations, user preferences, and institutional practices. These characteristics indicate that enterprise control tools need to be *adaptable* and *responsive to changes in context.*

Based on this insight we drew from our experiences with INSPECT and other systems to design a reusable system for critiquing and evaluating plans, which can be instantiated and applied to any planning domain, such as air campaign planning or army course of action critiquing. We used this as a platform to develop a knowledge acquisition tool called PSMTool that allows domain experts to enter new evaluation criteria for plans. PSMTool makes use of general ontologies that we developed for planning and plan critiquing to help users define and organize the criteria through a simple dialog. In preliminary experiments at Fort Leavenworth BCBL conducted through DARPA's HPKB project, we found that end users were able to use PSMTool with very little training to add new critiques that without the tool could only be added by a knowledge engineer. This system extends the approach of keeping human users in the loop and building tools that can easily be adapted in response to new conditions.

In this paper, we first describe the INSPECT system: the problem it solves, its design, architecture and implementation. We emphasize some of the lessons learned in this respect. Then we describe PSMTool and the further lessons from building such a highly adaptable tool. Finally we present our lessons learned for the contruction of enterprise control systems.

2 The Air Campaign Planning Domain and the Design of INSPECT

INSPECT was developed as a tool to support the air campaign planning process. The tool had to provide support to the adoption and use of a methodology for military planning called strategies-to-tasks [12, 11]. In this approach, high-level national objectives are refined into lower level

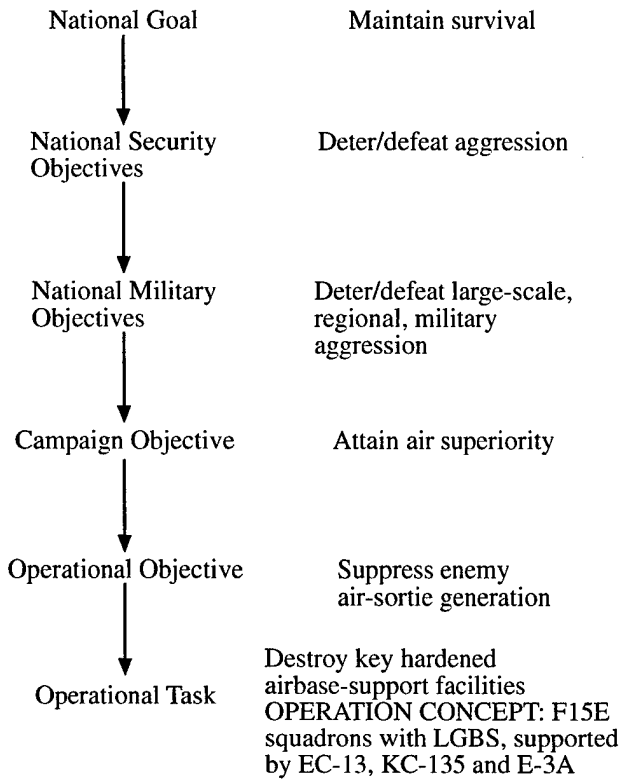


Figure 1. Strategies-to-Tasks Example (from [Todd 94])

military objectives and then into operational tasks. Figure 1 illustrates the hierarchy of objectives and how it links low level operational activities to higher level objectives. By encouraging planners to think about how low-level decisions relate to higher level objectives, this approach promotes air campaign planning that is more rational and helps avoid the sorts of mistakes that can arise from thinking about targeting too locally. The result should be air campaign plans that achieve the desired results while reducing risks and minimizing unnecessary damage. These plans are also more structured than traditional plans and capture more of the rationale behind the plan.

Our goal was to design an application that integrated well with this methodology and the air campaign planning process. Our domain experts had a number of useful suggestions about the design of the tool, most notably about issues regarding its interaction with the user. First, they requested that the user should not be locked in by the tool: the tool should present suggestions, not definitive answers, and users should be encouraged to think of alternatives. Second, they requested that all problems identified in the plan should be accompanied by an explanation and justification, so that the user could understand the reasoning behind the tool's recommendation and make an informed decision. Third, they asked that the problems should be presented as constructive criticism, including suggestions of possible fixes to the problem. Not surprisingly, following these suggestions allowed us to make our tool more attractive to users.

3 INSPECT: An Air Campaign Plan Evaluation Tool

Based on the design decisions described above, we developed INSPECT (INtelligent System for air campaign Plans Evaluation based on expeCT). The architecture of INSPECT is shown in Figure 2.

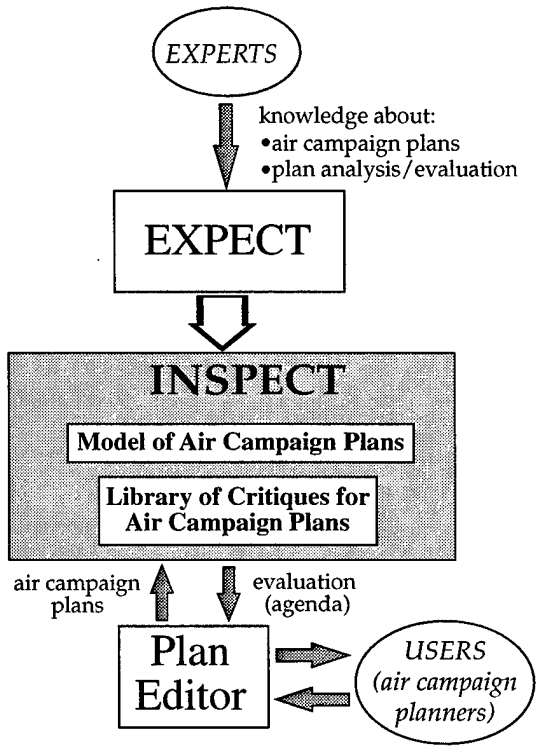


Figure 2. Architecture of INSPECT.

After entering air campaign objectives with a plan editor, a user invokes INSPECT to evaluate the plan. INSPECT looks for several types of problems in the plan. For example, it checks the hierarchical structure, identifies incomplete or incoherent objectives, and performs rough feasibility estimates based on the resources available for the campaign. INSPECT shows the user an agenda with all the problems found with the plan. The agenda items are marked according to their seriousness using a convention very familiar to Air Force pilots: WARNING, CAUTION, and NOTE (warnings requiring immediate attentions, and notes being non-critical). In addition to pointing out these problems, INSPECT can provide a detailed explanation of each problem and also suggest ways to fix it. Figure 3 shows a snapshot

of the agenda, including an explanation and suggested fixes for one of the agenda items.

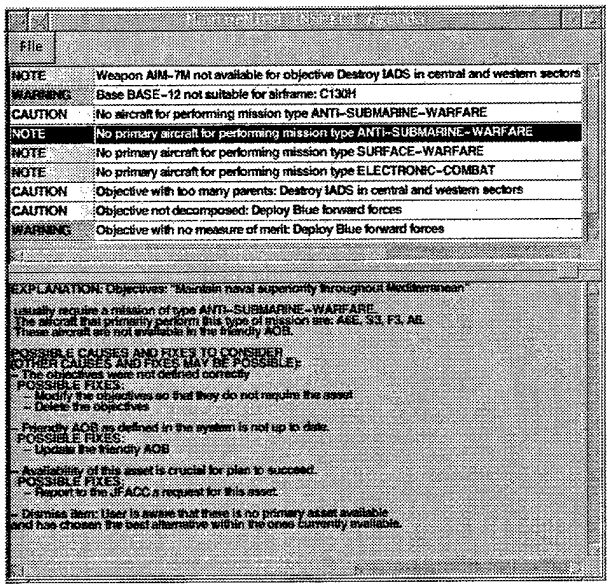


Figure 3. INSPECT's Evaluation of an Air Campaign Plan. The explanation (lower frame) refers to the selected item on the agenda (upper frame).

The types of problems detected by INSPECT include:

- *Objective with no child/parent.* According to the strategy-to-task approach to air campaign planning, all objectives must be subordinated to higher objectives and be decomposed further (until a pre-specified "leaf" level).
- *No objective fulfilling one of the basic tenets of air power.* Air Force doctrine suggests that an air campaign plan must contain objectives for all the tenets of air power, such as force deployment and force protection.
- *Objective with too many parents.* An objective with too many parents is an indication that the parent objective is either too general (and should thus be divided) or that the connections are meant to emphasize priority rather than reflect a true decomposition.
- *Incompatible sequence restrictions.* This problem occurs when temporal constraints of two or more objectives are contradictory.
- *No adequate aircraft currently available for an objective.* This problem occurs when an objective requires a type of mission for which none of the aircraft available is considered adequate.
- *Incoherent decomposition.* In principle, an objective must be decomposed into objectives which are more specific or more detailed than their parent. This problem occurs when a parent objective is subsumed by one of its child objectives.

There are a number of benefits for air campaign planning that stem from using a tool like INSPECT:

- **Catch errors introduced during manual plan development**: Air campaign plans are very large and complex, and need to be changed over time. Because the plan creation process is manual, it is conducive to introducing errors and inconsistencies in the plan. In fact, INSPECT has found unintentional errors in every plan generated by our domain experts which would otherwise have gone unnoticed.
- **Raise the floor on plan quality**: An evaluation tool like INSPECT helps users avoid creating inconsistent or low-quality plans. Because it works with the higher levels of the plan, it can detect problems that could percolate down to the lower levels and be accentuated as the plan details are worked out.
- **Enforce "good" practices in plan construction**: INSPECT's evaluations enforce the strategy-to-tasks methodology, and a number of style guidelines that experienced air campaign planners developed using it. Each of our domain experts liked to hear about the evaluation criteria suggested by other experts, as a new insight on how they could improve their own work on planning air campaigns. Our experts liked that INSPECT would point out that they were following their own standards as it evaluated their plans.
- **Training new planners**: INSPECT's knowledge base captures the knowledge of experienced planners, and novice users can learn as they use the tool. INSPECT's evaluations point out what experienced planners would see as serious flaws in the plan, and the explanations of each agenda item are designed to back up the evaluations with sources of information in the air campaign planning domain (doctrine, typical capabilities of weapon systems, etc.). INSPECT's suggested fixes show what an experienced planner would do differently.

4 Building INSPECT with EXPECT

INSPECT integrates several AI technologies. It was built using the EXPECT framework for knowledge-based systems development, that incorporates knowledge acquisition techniques, a description logic-based knowledge representation system, and a sophisticated problem-solving language and reasoner. EXPECT also has a language to express problem solving goals that is based on case grammars. Below, we briefly describe these technologies, and how they were used.

The knowledge acquisition bottleneck is frequently cited as a major impediment to broad dissemination of AI tech-

nology. The EXPECT project [6, 7] is addressing this problem by developing a knowledge acquisition framework that empowers people to augment, modify and adapt knowledge based systems without needing to understand the details of the system's implementation. The key to EXPECT's approach is that it captures the design rationale for knowledge based systems, and uses that design knowledge to guide a user in augmenting the system. In addition to INSPECT, EXPECT has been used to build several knowledge based systems in domains such as transportation planning and battlefield assessment.

Most knowledge acquisition tools have a fixed set of guidelines or expectations about how knowledge should be added to a system. The problem with this approach is that it is inflexible, and limits the range of systems that can be supported. EXPECT takes a more flexible approach: it automatically derives a knowledge-based system from abstract domain facts and problem-solving methods. The derivation process is recorded so that EXPECT captures the normally implicit dependencies in a KBS, such as what factual knowledge is needed to support problem solving, and how factual knowledge is used in problem solving. EXPECT provides tools that use this information to guide the user in adding knowledge and tools (such as a natural language explanation facility) that help make EXPECT's representations more understandable to non-computer experts. For example, the system understands how various types of instances are used in problem solving, so when a new instance is added the acquisition tools can make sure that enough information is specified about the instance so that it can be used. In this way, EXPECT allows a user to add knowledge to a knowledge-based system without requiring him to understand all the details of how the knowledge interacts.

The EXPECT system is fully integrated with the LOOM knowledge representation system [9]. LOOM is an implementation of description logics, which emphasizes efficiency and expressiveness instead of completeness. In EXPECT, LOOM is used to represent the factual and definitional knowledge about a domain. For example, in INSPECT there are LOOM definitions about what are the elements of air campaign plans, what are objectives, what are known types of aircraft, what kinds of missions they fly, etc. This knowledge has proved to be an important byproduct of the INSPECT development. It has been used as a basis for the development of a broad ontology of air campaign planning, which is being used and further developed under the JFACC DARPA Program.

INSPECT was built using EXPECT as follows. General knowledge about air campaign plans, their structure and contents, as well as general domain knowledge about air fight was coded into a LOOM knowledge base. Procedural knowledge on how to evaluate the plan according to the critiques specified was acquired and represented as EXPECT methods. The EXPECT system then put together these two types of knowledge, indicating whether there were any gaps or problems. The result of this process is an EXPECT model that records all the dependencies between procedural and domain knowledge. This model was then passed through the EXPECT compiler, that transformed it into efficient Lisp code that is able to solve the specified problem.

4.1 Representing Air Campaign Plans and Objectives

A very prominent contribution of our work resulted from integrating INSPECT with the plan editor tool. We designed a representation for air campaign plans and objectives that would allow both users and tools to exchange information about the plan. This representation has been adopted by other planning tools in the air campaign planning domain throughout the ARPI and JFACC programs, and is now seen as an important input to an ongoing effort in the US Air Force to create a common representation of objectives and tasks for air operations planning.

In integrating INSPECT with the plan editing tool (ACPT), we found a representation gap. Objectives in ACPT were represented with a sentence like "Gain air superiority in the western region", or "Destroy petroleum distribution facilities before the 15th day of the campaign". This was an unconstrained string, and the planner could write whatever came to his/her mind.

In order to be able to automate any interesting evaluation of the plan, we needed to capture the objective statement in a formal representation language. Parsing and interpreting the natural language sentence was too complex (and a new problem by itself). At the same time, the users were not willing to write their objectives in a form substantially different from the one they already used. The representation we proposed was therefore a middle-ground: we used a *case grammar*.[1] The basic idea of case grammars is that there is normally a limited number of roles (called *thematic* or *case* roles) that an argument of a verb can play with relation to the verb. That was definitely true for the objective statements, for several reasons. First, we found that the objective statements followed a very regular grammar of the form `<verb> <roles>`. Second, we found out that there were several regularities on the use of this structure. For example, only a handful of verbs (less than 30) are used. Third, each of these verbs introduces limitations with respect to the types of roles that can be used. For instance, most occurrences of the action type Destroy refer to a (physical) object type like "missile launch sites" or "military headquarters". We were

[1] While case grammars have been dismissed as a general solution for natural language interpretation, they can be an interesting and powerful device in restricted settings such as the one we have in the air campaign planning domain.

able to establish reasonably exhaustive lists of terminals for each of the main types specified for role fillers. Fourth, we found that certain roles were actually *modifiers* that are used to specify restrictions or constraints on the objective. There are three types of restrictions, for time (e.g., within 21 days), space/area (in Western Region) and resources (using B-52s from base XYZ). A diagram showing the structure of the proposed grammar is shown in Figure 4.

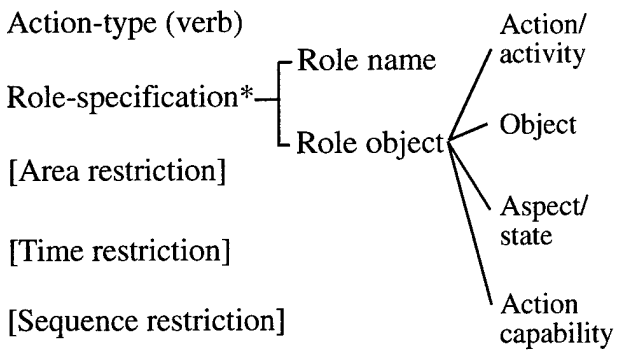


Figure 4. Overall structure of the case grammar to represent air campaign objectives.

There were several benefits to this representation. On the systemic side, it allowed the integration of ACPT and INSPECT. A syntax-oriented editor was built that helps the users enter valid sentences by offering lists of valid completions (according to the grammar) for the text being entered. This provides a proactive support for using for the grammar in the edition of objectives, without unnecessarily constraining the planner, who still has the liberty to write free text if he/she deems necessary. The case grammar became a shared representation that allowed other applications to make use of the more semantic representation.

Somewhat unforeseen were a number of methodological benefits, i.e., the benefits of using a grammar to the planning process itself, independently of any tools used. These benefits were so important that the structured representation took a life of its own and is often seen as a key contribution of our work on INSPECT. First, the knowledge acquisition process involved in building the representation forced the experts to explain and reflect over the way they write air campaign objectives. For instance, they came to the conclusion that frequently occurring objectives like "Conduct operations" should not be allowed because they in fact do not mean anything — in a air campaign plan, basically everything can be seen as conducting operations. Second, the resulting grammar embeds the notion of "reasonable" objectives, which had never been made explicit before then. Third, the additional structure provided by the grammar was considered particularly useful for training. Fourth, the case grammar became an input to an ongoing standardization process in the Air Force on specifying valid types of tasks and objectives. Indeed, the success with this representation has led us to participate in the development of specialized representations for other elements of air campaign plans, as well as in extending the existing representation of objectives.

5 Adding new critiques with PSMTool

INSPECT supports the user in simple modification and maintenance tasks by virtue of the underlying EXPECT system. However we wanted to provide support for adding new critiques to a critiquing tool such as INSPECT. From our experiences with INSPECT as well as critiquers for logistics planning [8] and army courses of action [2] we observed that plan critiques often follow one of a set of generic patterns which could be captured using an ontology of planning and critiquing. This ontology can be used by a computer program to provide guidance for adding new critiques through dialogue with users. It can also help to organize the critiques and give a baseline estimate of the completeness of the critiquing system, all of which helps to increase user acceptance of the plans that are generated. More details about the ontologies and their coverage of real-world critiquing tasks can be found in [2].

We implemented a knowledge acquisition tool called PSMTool that uses these principles to acquire new critiques from users. The tool is domain-independent but requires that the domain has been aligned with the generic ontology of planning and critiquing, which it uses to express the new critiques. Even with this tool, specifying a new critique involves specifying problem-solving knowledge, so PSMTool also makes use of an editor developed to enter problem-solving knowledge for the Expect system using English-like syntax [4]. We tested PSMTool at Fort Leavenworth with the help of the Battle Command Battle Labs and found that Army officers with very little training were able to add significant new critiques to an Army battle course of action critiquer. In this section we describe the design and implementation of PSMTool and briefly describe the results of the user tests.

PSMTool uses two ontologies to represent general plan critiquing strategies: an ontology of plans and an ontology of critiques. Generic problem-solving knowledge is attached to the critiques as we now describe. The planning ontology, called Planet [1], was developed under the Darpa-ran HPKB project and is a general ontology that allows both machine-generated and human-generated plans to be represented and also explicitly represents different assumptions made by planners. It has also been used as the basis of a translation service for software agents collaborating on a planning task [3].

The critiquing ontology represents domain-independent critiques of two types: those based on the structure of the

plan and those based on its use of resources. Consider, for example, the problems detected by INSPECT that are described in the previous section.

- The critiques *objective with no child, objective with too many parents* and *objective with incompatible sequence restrictions* are based on the structure of the plan. They are independent of the air campaign domain and the first two are included in our ontology of critiques along with the problem-solving knowledge required to implement them.
- The objective *incoherent decomposition* is also based on plan structure but requires domain knowledge to know if a parent objective is subsumed by one of its parents. In this case the critique ontology provides support for evaluating a set of objectives with respect to their parents, and the details are fleshed out as part of the domain definition.
- The objective *No adequate aircraft currently available for an objective* is a resource-based critique that requires domain knowledge to implement. Again, the critiquing ontology provides partial support.

PSMTool uses these ontologies to classify a new critique added by the user in a question-answering process. Once the critique is classified in the ontologies, the appropriate generic problem-solving knowledge can be applied and the domain-dependent knowledge that is required can be identified. In many cases, this process also breaks down the knowledge that needs to be acquired into small chunks that are easier for a domain expert to express.

Figure 5 shows the PSMTool interface while a new critique is being added. This is a critique from the army course of action domain that checks that the friendly forces have enough force ratio for each of the tasks in the battle, according to standard practice. The window on the left shows how a three-part script is being followed to add the critique. In the first part, four questions were answered that allowed PSMTool to classify the critique, showing that it is a quantity-based check made on each task in the plan. In the second part, PSMTool explains how the critique will be implemented based on this classification, and identifies pieces of problem-solving knowledge to acquire from the user to complete the critique. In the third part, which has not yet been reached, PSMTool will run the critique on an existing course of action so the user can check the results.

In the case of the force ratio critique, PSMTool asks for two pieces of problem-solving knowledge: how to estimate the amount of force ratio required for a task and how to estimate the amount available for a task. In the right-hand window in Figure 5, the English-based editor is being used to specify how to estimate the amount of force ratio available to a task. More details about the editor, how it produces an English paraphrase of the procedure body and a set of alternatives that allow users to create and modify procedures through navigation can be found in [4].

6 Experiences with PSMTool

Preliminary experiments were run with PSMTool at Fort Leavenworth. Four subjects who had been given one day of training with Expect and the representation of the course of action (COA) domain were presented with four new critiques to add to the system. The subjects were army officers who were reasonably familiar with the COA generation and whose use of computers ranged from reading email to having been exposed to java. Two of the critiques were added using PSMTool and two without the tool. In order to minimize learning effects on our results, two of the subjects worked without PSMTool for two critiques and then worked with the tool, and two of the subjects worked the other way around.

Our results, although preliminary, show that not only can users add more critiques using the tool than without, they are also faster and less error-prone when using the tool. Moreover, the subjects reported the critiques that they added using the tool as being simpler, even though they were in fact isomorphic. This provides some support for the idea that, by classifying the new critiques with the ontologies as they are added, PSMTool helps users to organize the critiques in their own minds.

In a second experiment we tested whether PSMTool could be used with little or no training. A fifth subject, an army officer with no programming experience, was asked to add two of the same critiques as in the previous experiment, but without the day of training in Expect and the domain, instead the tool was demonstrated and explained for approximately 45 minutes. The subject successfully added the critiques, in time comparable with that of the earlier subjects who had received training.

We are currently designing more thorough user experiments to test these highly encouraging results. However they seem to indicate that the goal of allowing subject matter experts to add new evaluation criteria to a plan critiquer by directly interacting with an automated system is attainable, even if the experts have little training with the system. If this is the case, we can hope to greatly increase the range of scenarios in which an enterprise control system can be usefully applied, by allowing it to be modifiable by its users.

7 Lessons Learned: Mixed-Initiative Tools for Enterprise Control Systems

In order to obtain cost savings and scalability, the ideal scenario for enterprise control systems seems to argue for a completely automated control system. However, our experience with INSPECT has shown that in large-scale, real-world

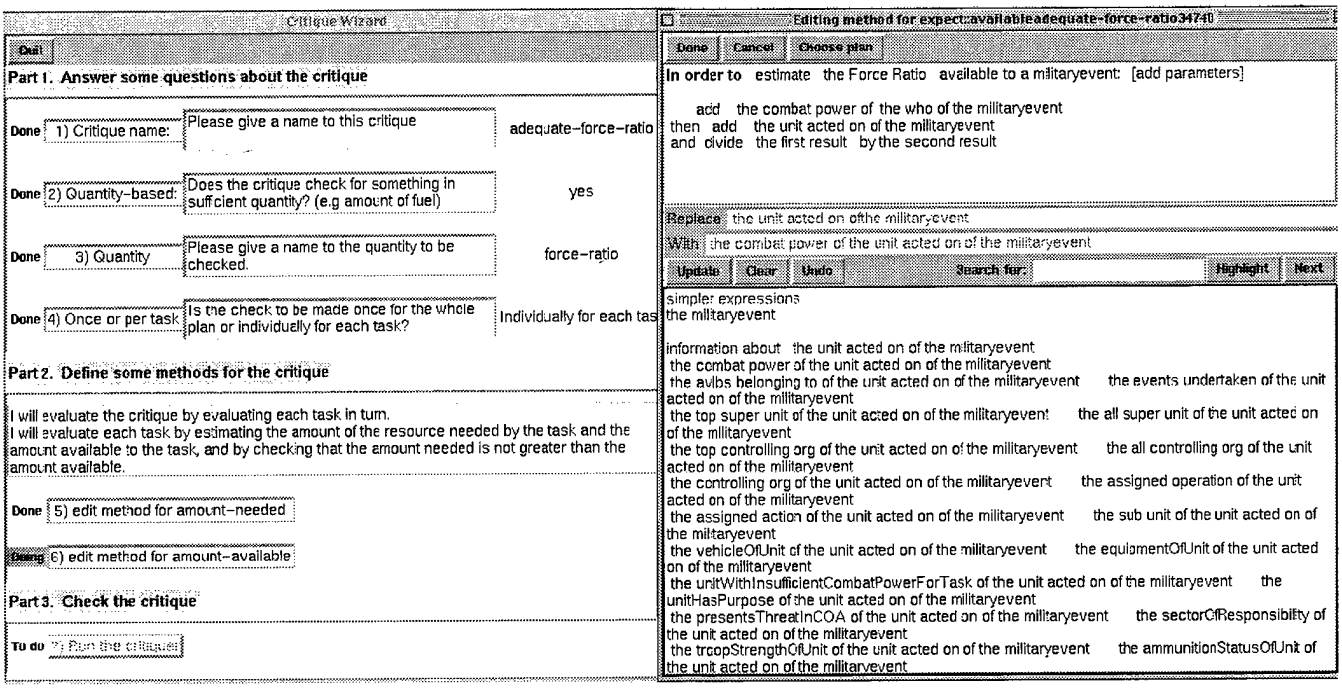


Figure 5. Adding a new critique with PSMTool. The window on the left shows the questions that were asked to define the critique and the new knowledge that is required. In the window on the right, some of that knowledge is entered using the English-based editor.

domains, completely automated control is often impracticable due to the scope and breadth of knowledge required. We are not alone with this view. For example, [5, 10] argue that for planning — a key task in enterprise control — a mixed-initiative approach where machines and people work together is often more desirable.

The key insight is that people can understand the enterprise problems and their *context* more broadly than machines, and thus will be able to make better judgments about certain decisions. Machines, on the other hand, are able to carry out tasks with a well-defined context much more effectively. Another issue is *adaptability*: humans are able to understand that their knowledge about a certain kind of control process is no longer valid and seek to revise this knowledge at the light of the new information. Machines simply do not have that capability at the moment, and thus behave with brittleness.

Morover, even if we do develop techniques that can overcome these limitations, it is an open question whether giving all the power to the machine is a desirable choice. Humans tend to be afraid of giving up control on certain key decisions, and particularly so when they do not have access to an explanation as to why the decision was made, what other decisions were possible, and what are other choices. Black-boxes are simply not acceptable for many key control decisions, particularly in applications such as military air campaign planning, where lives are at stake.

As a result of this view on enterprise control systems, we argue that these systems should adopt a *mixed-initiative approach* with two main characteristics. First, they must be designed to work with people, rather than completely taking over processing. This in turn means that the products of our tools must be *understandable* by people and it must be possible for people to easily input information and decisions into the tools. Second, because it is impossible to anticipate in advance all the knowledge a system might need in a broad domain, and because knowledge frequently changes, our goal is to provide *knowledge acquisition tools* that allow for users to augment and adapt a system's knowledge in response to new situations and new needs.

Acknowledgments

The work reported here relates to research sponsored by the Defense Advanced Research Projects Agency (DARPA) under Ft. Huachuca Contract DABT63-95-C-0059 and Air Force Research Laboratory Agreements F30602-97-C-0118 and F30602-97-C-0068. Many thanks to all Checkmate members who helped us in the process; most Col Plebanek, who allowed us to have this interaction, LtCol Cardenas, who coordinated the knowledge acquisition sessions and

was a central expert, and the experts we worked with more closely: Maj Allison, LtCol Alred, LtCol Cardenas, Maj Cunico, and Maj Jackson. We are also grateful to the staff of the Battle Command Battle Labs at Fort Leavenworth for their assistance in running user experiments, especially Capt Rasch and Col Duquette. The views and conclusions contained in this article are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

[1] J. Blythe and Y. Gil. Planet: A shareable and reusable ontology for representing plans. Technical report, Expect internal report, 1999.

[2] J. Blythe and Y. Gil. A problem-solving method for plan evaluation and critiquing. In *Proc. Twelfth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, 1999.

[3] J. Blythe, Y. Gil, H. Chalupsky, and R. MacGregor. Supporting translation among planning agents. In *Submitted to the Fifth International Conference on Artificial Intelligence Planning Systems*, 2000.

[4] J. Blythe and S. Ramachandran. Knowledge acquisition using and english-based method editor. In *Proc. Twelfth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, 1999.

[5] G. Ferguson, J. Allen, and B. Miller. Trains-95: Towards a mixed-initiative planning assistant. In B. Drabble, editor, *Proc. Third International Conference on Artificial Intelligence Planning Systems*, University of Edinburgh, May 1996. AAAI Press.

[6] Y. Gil and E. Melz. Explicit representations of problem-solving strategies to support knowledge acquisition. In *Proc. Thirteenth National Conference on Artificial Intelligence*. AAAI Press, 1996.

[7] Y. Gil and B. Swartout. Expect: Explicit representations for flexible acquisition. In *Proc. Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, 1995.

[8] Y. Gil and W. R. Swartout. Expect: A reflective architecture for knowledge acquisition. In *Proceedings of the 1994 Workshop of the ARPA-Rome Laboratory Knowledge-Based Planning and Scheduling Initiative*, Tucson, AZ, February 1994.

[9] R. MacGregor and R. Bates. Inside the LOOM description classifier. *SIGART Bulletin*, 2(3):88–92, June 1991.

[10] K. Myers. Strategic advice for hierarchical planners. In *Proceedings of the International Conference on Knowledge Representation*, 1996.

[11] D. Thaler. Strategies to tasks, a framework for linking means and ends. technical report, RAND Corporation, 1993.

[12] D. Todd. Strategies-to-tasks baseline for usaf planning. Internal document, Strategic Planning Division, HQ United States Air Force, 1994.

Section 3

Optimal and Adaptive Control

A Framework for the Decentralized Control of Manufacturing Enterprises[1]

Michael C. Caramanis
mcaraman@bu.edu

Ioannis Ch. Paschalidis
yannisp@bu.edu

Osman M. Anlı
anli@bu.edu

Boston University, Department of Manufacturing Engineering, Boston, MA 02215

Abstract

We present a framework for synergistic and decentralized decision making in complex stochastic systems and related research findings. The framework is related primarily to manufacturing systems and associated supply chains, but is also relevant to other fields such as economics (complex organizations), computational physics (fluid mechanics), and communication systems. We report on selected findings of an interdisciplinary research effort supported by a Knowledge and Distributed Intelligence (KDI) award by the National Science Foundation (see `http://www.bu.edu/pcms/kdi/`), and present results on vertical and horizontal coordination of manufacturing supply chains.

Keywords: Decentralized Control, Supply Chain Coordination, Hierarchical Decomposition, Enterprise Integration.

1 Introduction

1.1 A Synergistic Decentralized Decision Framework

The size, nonlinearity, and stochasticity of manufacturing systems renders effective (let alone "optimal") centralized decision making intractable. To make a more precise argument we define a manufacturing system's state $x(t)$, randomness $w(t)$, decision vector $u(t)$, cost rate $g(x(t), u(t), w(t))$ and system dynamics equation $\dot{x}(t) = f(x(t), w(t), u(t))$. The size of $x(t)$, the complexity of f, and the stochastic nature of $w(t)$ – size, observability, span of a range of characteristic frequencies etc. – do not allow tractable formulation of optimal decision rules of the type $u(t) = \mu^*(x(t))$ that minimize:

$$\bar{J}_\mu = \mathbf{E}g((x(t), u(t), w(t)),$$
$$\text{subject to } \dot{x}(t) = f(x(t), w(t), u(t)).$$

In fact, a centralized decision approach is not only intractable when applied to very large multi-factory enterprises; the difficulty persists even when subsystems, such as a single factory, are considered.

A practical, yet arguably inefficient approach adopted in practice has been to decompose the decision vector $u(t)$ to decoupled components (in the example of Figure 1, $u(t) = [u_1(t), u_{2,c}(t), u_{3,p}(t), \ \forall c, p]$) and assign them to decentralized decision entities (e.g., individuals or teams). The responsibility of these entities is to establish decision rules of the type $u_{\ell,i}(t) = \mu_{\ell,i}(\phi_{\ell,i}(x(t)))$, where the pair (ℓ, i) identifies the decision entity (by aggregation level ℓ and node index i defined below) and $\phi_{\ell,i}(x(t))$ denotes the subset of the system state information made available to, and used by, the decision entity. This practice produces tractable but inefficient decision rules because (i) it underestimates the interaction of decentralized decisions, and (ii) does not model uncertainty and salient system state dynamics. For example, an hourly scheduling strategy at the cell level that is not responsive to weekly production targets set at the factory level or to the expected repair time of a failed machine, will be quite inefficient. Furthermore, the average performance of such a scheduling strategy will misinform factory level planners about cell capabilities and induce them to plan under the perception of a lower available capacity. An analogous example in the context of a military enterprise is to develop operational strategies that are not conditional upon cloud cover forecasts and to estimate the probability of winning a battle by averaging over weather conditions when it is well known that decisions to go into battle are dependent upon the weather forecast.

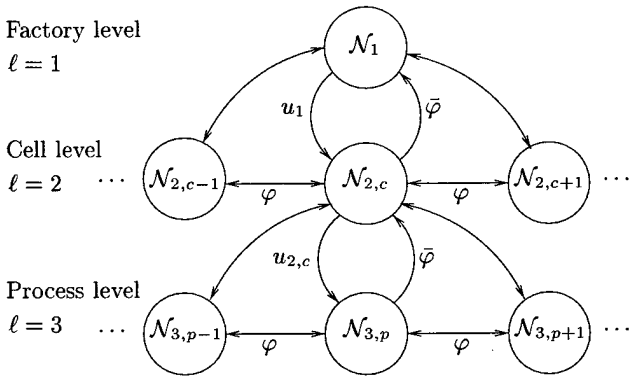


Figure 1: The information flow between decision nodes.

Our research attempts to transcend state of the art production management tools by achieving systematic coordination of decentralized decision entities. This coordination allows us to model uncertainty effectively and respond to available information in a timely fashion. We associate decision entities with a pyramid-like network of *decision/information nodes* that belong to the same or different *aggregation levels* (e.g., plant, manufacturing cell, and manufacturing process, shown in Figure 1 as levels $\ell = 1, 2, 3$). The aggregation level, ℓ, defines the scope of the decisions made by nodes located at that level. Nodes at high aggregation levels are re-

[1]Research supported by the NSF under grants ACI-9873339 and NCR-9706148, by the Alcoa Research Foundation, and the Nokia Research Center (Boston).

sponsible for decisions with a wider scope that impact a larger group of resources. For example, a node at the factory level allocates machines and weekly production targets to all manufacturing cells in the plant, while a manufacturing cell node makes decisions that affect directly the management of machine stations within the cell. A parent node's decisions affect all resources impacted by decisions made at its descendant nodes. However, parent and descendant node decisions affect resources in different ways and are made at different frequencies. For example the plant level node assigns machines to manufacturing cells every year, while a cell level node decides on daily production. We associate nodes with decisions *and* information because there is a clear tradeoff between the scope of the decisions assigned to a node and the accuracy with which the capabilities, dynamics, stochasticity, and performance of the affected resources need to be represented at that node. Present practice of manufacturing system management is characterized by the adoption of a rudimentary "coordination technology", and as such it is far from efficient. We believe that it is feasible to increase information and modeling fidelity without compromising tractability. The new paradigm of synergistic decentralized decision making exploits time-scale directed decomposition opportunities to promote tractable decision rules of the form

$$u_{\ell,i}(t^\ell)=\mu_{\ell,i}\Big(\phi_{\ell,i}(x(t^\ell)), u_{\ell-1}(t^{\ell-1}), w_{\ell-1}(t^{\ell-1}), \varphi_{\ell,j}(t^\ell), \bar{\varphi}_{\ell,i}(t^{\ell+1}) \ \ \forall j\Big),$$

where t^ℓ represents time evolving according to the time scale of level ℓ, and $w_{\ell-1}, u_{\ell-1}, \varphi, \bar{\varphi}$ represent a state augmentation with the following additional information made available to node (ℓ, i) (see Figure 1):

- The parent node provides node (ℓ, i) with (i) uncertainty realization $w_{\ell-1}$ which is static during the finer time scale periods $t^{\ell+1}$ of descendant node dynamics that are subdivisions of t^ℓ and (ii)a directive or target which is in fact the parent node's decision $u_{\ell-1}$. Since the parent node makes decisions much less frequently than the descendant node, $u_{\ell-1}$ can be treated as static for purposes of decision making at node (ℓ, i).

- Nodes at the same level, ℓ, provide sufficient statistics, φ, of their own decisions for horizontal coordination purposes. Since same level nodes make decisions at similar frequencies, the statistics will not be time-averaged, and will have the form of a probability distribution or the sufficient statistic (e.g., distribution or moments of the demand process for the upstream node provided by the downstream node).

- Nodes at lower levels, i.e., descendant nodes, provide time-averaged statistics, $\bar{\varphi}$, of their own decisions and the resulting performance for vertical coordination purposes. Time averaging is needed for consistency, since descendant node decision time scales are smaller. It is often beneficial for these statistics to include sensitivity estimates of performance with respect to the directives $u_{\ell,i}(t^\ell)$ passed by node (ℓ, i) to its descendants. Furthermore, the time averaged statistics must be conditional upon uncertainty realization and other information which is static for the duration of each coarse time unit in the dynamics of the parent node $\ell - 1$.

There is no doubt that synergistic decentralized decision making has a tremendous potential to improve the efficiency of complex stochastic systems. It is therefore remarkable that despite the availability and affordability of computation, information gathering, and communication resources, manufacturing enterprise integration is far from being a reality. In our opinion, one of the reasons is that raw computational power – which exists in abundance – alone does not suffice to provide the appropriate tools without the incorporation of "intelligence". To this end, the research community must (i) develop new computational and algorithmic capabilities required at individual nodes to exploit information from other nodes and to generate information needed by other nodes, and (ii) build on the theory of decentralized control and time-scale-driven decomposition [1, 2, 3, 4] to develop a science base for designing the requisite vertical and horizontal information exchange between decision/information nodes, namely identify the desired information that summarizes the behavior at lower levels for use at higher levels and characterize consistent rules for horizontal information exchange across same level nodes.

1.2 Effective State and Dynamics in a Factory-Cell-Process Decomposition Example

Consider for illustration purposes the three aggregation levels of Figure 1. The plant node is denoted by $\mathcal{N}_1$, while the cell and process nodes are denoted by $\mathcal{N}_{2,c}$ and $\mathcal{N}_{3,p}$ to identify their level (2 or 3) and position with each level (cell c or process p). The proposed synergistic and decentralized decision making approach utilizes an information sharing framework that enables computation of tractable and efficient decision rules. These rules capture real-time dynamics and uncertainty in a way that substantially closes the gap between intractable centralized optimal decisions and today's static inefficient practice of decentralized decision making. Tractability is achieved at minimal deviation from optimality by introducing three types of distributed information: (i) State aggregation represented by the functions $\phi_1(x(t^1)), \phi_{2,c}(x(t^2)), \phi_{3,p}(x(t^3))$, (ii) Decision aggregation without time-averaging (i.e., generated and used by same level nodes) represented by the functions $\varphi_{2,c}(t^2)$, conveying information from node $\mathcal{N}_{c'}$ to node $\mathcal{N}_c$, and (iii) Decision and performance statistics with time-averaging (i.e., generated at a descendant node and used by a parent node) represented by the functions $\bar{\varphi}_1(t^2)$ and $\bar{\varphi}_{2,c}(t^3)$.

Tractable decision rules can now be developed by exploiting decision frequency differences across aggregation levels. The form of the proposed decentralized synergistic decision rule is:

$$u_1(t^1) = \mu_1\Big(\phi_1(x(t^1)), \bar{\varphi}_1(t^2)\Big),$$

$$u_{2,c}(t^2) = \mu_{2,c}\Big(\phi_{2,c}(x(t^2)), u_1(t^1), w_1(t^1), \varphi_{2,c}(t^2), \bar{\varphi}_{2,c}(t^3)\Big),$$

for the plant and the cell level, respectively. A similar rule applies for the process level.

The dynamics for each node can now be decomposed. Using the subscript convention to represent random variable decomposition and suppressing double arguments we can write:

$$\dot{\phi}_1(t^1) = f_1\Big(\phi_1(t^1), \bar{\varphi}_1(t^2), u_1(t^1), w_1(t^1)\Big)$$

$$\dot{\phi}_{2,c}(t^2) = f_{2,c}\Big(\phi_{2,c}(t^2), u_1(t^1), w_1(t^1), \varphi_{2,c}(t^2), \bar{\varphi}_{2,c}(t^3), u_{2,c}(t^2), w_{2,c}(t^2)\Big)$$

The difference in the frequency of decisions is crucial. Long time scale decisions and stochastic events can be treated as static at lower (i.e. shorter time scale) decision levels. Shorter time scale decisions and stochastic events can be summarized at higher (i.e. longer time scale) decision levels via the statistics of their resulting performance, i.e., the $\bar{\varphi}$ functions. In the vertical and horizontal information sharing problem of manufacturing cell coordination considered in Section 2 and Section 3, the objective of the decentralized decision framework is to minimize inventory and backlog costs across all cells by utilizing the following decisions and information sharing:

• At the *factory level*, decisions $u_1(t^1)$, include resource allocation and part type assignment to cells (group technology), and weekly production targets to cells (production planning). State information, $\phi_1(x(t^1))$, includes the pool of available resources and the weekly delivery requirements of final product types. Cell level statistics, $\bar{\varphi}_1(t^2)$, are used for plant level decisions but are generated at cell level nodes; they include achievable regions in the average weekly production and *lead time* space for each cell and associated sensitivities. The cell level nodes generate these statistics. In Section 2 where vertical cell coordination is considered through the interaction of factory and cell level nodes, $u_1(t^1)$ denotes decisions of a fluid model at the factory level on part type and time bucket specific production requirements of interacting cells, while $\bar{\varphi}_1(t^2)$ denotes average production and sensitivity statistics generated by cell level nodes to capture the requisite information about cell specific lead times as a function of cell production requirements.

• At the *cell level*, decisions, $u_{2,c}(t^2)$, include labor and resource allocation to workstations (layout), material release policies, workstation loading schedules and part-type routing schedules. Targets from the plant level are plant level decisions u_1. Horizontal information, $\varphi_{2,c}(t^2)$, is generated by and exchanged between interacting cell nodes. This information has a probability-distribution-type content of short term (i.e., relative to a far shorter time unit than the coarse time unit used in the factory level production plan) required release and production rates of cells. Manufacturing process statistics, $\bar{\varphi}_{2,c}(u_{3,p}(t^3))$, are processing time and quality statistics provided to node $\mathcal{N}_{2,c}$ by all process level nodes $\mathcal{N}_{3,p}$ which are descendants of node $\mathcal{N}_{2,c}$. In Section 3 where horizontal cell coordination is considered through the interaction of cell level nodes, $\varphi_{2,c}(t^2)$ denotes distributional information on the production and release (i.e. demand) processes of each cell. This captures the salient characteristics of the possibly auto correlated stochastic demand and production processes of interacting downstream and upstream cells.

1.3 Supply Chain Coordination Issues

Supply chain coordination involves a multitude of decisions by various agents. As shown in Table 1, decisions are characterized by (*i*) varying scope – i.e affect directly larger or smaller subsets of supply chain components – (*ii*) time scale – for example yearly investment decisions, monthly group technology and layout modifications, weekly production planning decisions, hourly production operations decision – and (*iii*) functionality – for example resource allocation, uncertainty contingency planning, and sequencing. The characteristic time scale of average decision frequency (and the associated subsystem dynamics) decreases in the south east direction of entries in Table 1.

	Enterprise	**Plant**	**Cell**
Resource Allocation	Hardware Investment, New Product Development, Cost,Quality and Service Trade-offs	Group Tech. Cell Design, Assigning Machine & Labor to Cells	Facility Layout, Asgn.Mach.& Labor to W/S, Asgn.Preventive Maintenance Resources to Machines
Contingency Planning	Strategic Risk Planning, Alliances & Mergers, Diversification, Demand Variability Analysis	Safety Stock Analysis, Securing Farm Out Capacity to handle Uncertainty	Perf.&Sens. Analysis, Stochastic Scheduling Policy Design, Horizontal Cell Coord.
Sequencing	Scheduling of Plants	Master Production Planning, Set-up Run Scheduling	Detailed Machine Scheduling, Set-up Scheduling

Table 1: Example of decisions by functionality and scope.

Even if all components of the supply chain are subject to common ownership – such as cells of the same factory – centralized decision making is practically impossible due to the formidable complexity posed by stochasticity and dynamics with widely differing time scales. Decentralized decision making has been traditionally adopted to deal with this complexity. However, coordination has been limited by high cost and the lack of an appropriate science base for (*i*) the design of information exchange requirements, (*ii*) the assignment of decisions to autonomous decision agents, and (*iii*) the requisite algorithms. With the dramatic decrease in data gathering and communication costs, there has been ample of activity towards the creation of a science base that can promote coordination and increase productivity through effective supply chain coordination [2, 5].

In the remainder of this paper we focus on the *vertical* and *horizontal* coordination of planning and operational decisions of manufacturing cells (or focused factories) interacting either under common ownership or under a multiple-tier-supplier relationship. Production planning decisions involve medium term resource scheduling, such as weekend overtime shifts, and are characterized by a coarse time scale which we will refer to henceforth as the *time period* or *time bucket* (usually covers a week long or similar duration period). In contrast, production operation decisions, such as detailed daily production scheduling, response to unplanned machine failures, and short term *horizontal* coordination with upstream and downstream cells are

characterized by a finer time scale (usually a shift or an hour).

The *vertical* coordination approach presented in Section 2 is achieved by determining *optimal* and *consistent* production targets for each cell and part type during each time period in an appropriately selected planning horizon. Optimal refers to the minimization of expected Work- In-Process (WIP) and Finished-Goods-Inventory/Backlog (FGI/BL) costs across all supply chain components (cells) for a given forecast of final demand. Consistent refers to the achievability of production targets by all cells during all time periods, namely, assuming that every cell other than cell c meets its production targets, then material flow balance will allow cell c to meet its production targets without violating, capacity, lead time, and average safety stock between cells that results from the horizontal cell coordination policies presented in Section 3.

Cell coordination can be enhanced through *horizontal* information sharing between cell nodes over the finer time scale of cell dynamics. Indeed, the prevention of starvation from inadequate supply by an upstream cell can be achieved by determining minimal levels of safety stock targets in between cells. These targets can be thought of as dynamic kanban levels that vary over the coarse time scale of the factory level, and, as such, represent a time aggregated statistic determined at the cell level. Section 3 outlines a large deviations approach to meeting this requirement.

2 Vertical Cell Coordination

The state of the art of current practice in production planning adopts techniques which determine production targets by either ignoring some of the consistency constraints or by making them tighter so that they become linear and time invariant. The most common incidence of this approach is the use of a constant lead time – or worst case lead time, i.e associated with the highest allowed loading – in Material Requirements Planning (MRP) systems. The benefit of this simplification is the elimination of extensive coordination and information exchange with the operational decision making level. However, this benefit is unfortunately accompanied with significant loss of efficiency in terms of inventories, production delays, and final demand service rates [6, 7, 8].

We demonstrate how the proposed framework can capture the value of variable lead time information in the coordination of the supply chain and in particular the potential contribution of this information towards decreasing inventory and backlog costs. We investigate the computational burden associated with generating the requisite information through calculations distributed to the various components of the supply chain and communicated to a master-problem-solving coordinator. In particular, we study master-problem-solving requirements and performance for non-linear lead time relationships that are concave and relatively easy to model, as well as non-concave and harder to deal with. Computational experience is provided to elaborate on the value of variable lead time information and the effectiveness of the proposed master-problem-solving algorithms. Horizontal coordination constraints such as safety stocks and Kanban levels between supply chain components are not considered here for purposes of brevity and clarity of exposition of the basic coordination and value of information principles.

2.1 Problem Formulation

Consider a collection of cells $i \in \{1, 2, \ldots, i, \ldots, \mathcal{I}\}$ that represent components of the supply chain of interest. Each cell processes part type $r = 1, 2, \ldots, \mathcal{R}_i$, and contains work stations $M_{i,m}, m = 1, 2, \ldots, \mathcal{J}_i$ with buffers $q^r_{i,m}$.

Define:

$\overline{X}^r_i(t_k)$: the production target of part r by cell i during period t_k,

$Q^r_i(t_k) = \sum_m q^r_{i,m}(t_k)$ the work in process of part type r in cell i *at the end* of period t_k,

$\overline{Q}^r_i(t_k) = \alpha Q^r_i(t_{k-1}) + (1-\alpha) Q^r_i(t_k), \quad 0 \le \alpha \le 1,$ The average work in process of part r in cell i *during* period t_k,

$I^r_i(t_k)$: the finished goods inventory or backlog of part type r after cell i *at the end* of period t_k,

$\overline{R}^r_i(t_k)$: the release rate of part type r during period t_k into cell i,

$LT^r_i(t_k)$: the lead time or system time of part type r in cell i consistent with the loading of cell i during period t_k, and

$d^r(t_k)$: end product demand for part type r during period t_k.

For clarity of exposition we define an incidence of the general problem which is trivially generalizable. The example depicted in Figure 2 is a supply chain consisting of two subassembly fabrication cells feeding the final assembly cell. Three different part types are processed at three work stations in each cell. The Production Planning problem is then defined as the minimization of WIP and Finished Goods Surplus/Backlog costs over the planning horizon $\{t_1, t_2, \ldots, t_k, \ldots, t_K\}$, subject to material flow relations (*f1-f3*), capacity constraints (*c*), lead time constraints (*lt*), equality constraints (*eq*), and positivity constraints (*pos1-pos2*).

$$(of) \quad \text{minimize} \sum_{i,r,t_k} \left[w_{i,r} Q^r_i(t_k) + c^+_{i,r} I^+_{i,r}(t_k) + c^-_{i,r} I^-_{i,r}(t_k) \right]$$

subject to the following constraints enforced for $\forall r$ and $t_k \in \{t_1, t_2, \ldots, t_K\}$

$$(f1) \quad Q^r_i(t_k) = Q^r_i(t_{k-1}) + \overline{R}^r_i(t_k) - \overline{X}^r_i(t_k) \qquad \forall i$$

$$(f2) \quad I^r_i(t_k) = I^r_i(t_{k-1}) + \overline{X}^r_i(t_k) - \overline{R}^r_3(t_k) \qquad i = 1, 2$$

$$(f3) \quad I^r_i(t_k) = I^r_i(t_{k-1}) + \overline{X}^r_i(t_k) - d^r(t_k) \qquad i = 3$$

$$(c) \quad \sum_r U^r_{i,m}(t_k) \overline{X}^r_i(t_k) \le \eta_{i,m}(t) \qquad \forall m \in M_i, \ \forall i$$

$$(lt) \quad \sum_{k=1}^{k^*} \overline{R}^r_i(t_k) + \hat{Q}^r_i \ge \sum_{k=1}^{k^*} \overline{X}^r_i(t_k) + \sum_{k=k^*+1}^{K} \overline{X}^r_i(t_k) \mathbb{1}\{t_k - LT^r_i(t_k) \le t_{k^*}\} \qquad \forall i, \ k^* = 1, 2, \ldots, K$$

$$(eq) \quad \overline{Q}^r_i(t_k) = \alpha Q^r_i(t_{k-1}) + (1-\alpha) Q^r_i(t_k) \qquad \forall i$$

$$(pos1) \quad \overline{X}^r_i(t_k), \overline{Q}^r_i(t_k), Q^r_i(t_k), \overline{R}^r_i(t_k) \ge 0 \qquad \forall i$$

$$(pos2) \quad I^r_i(t_k) \ge 0 \qquad i = 1, 2$$

$$I^r_i(t_k) \text{ unrestricted in sign} \qquad i = 3$$

where:

$I^+_{i,r}(t_k) = \max\{0, I^r_i(t_k)\}$ positive finished goods inventory at the end of period t_k

$I^-_{i,r}(t_k) = \max\{0, -I^r_i(t_k)\}$ backlogged demand at the end of period t_k

$\hat{Q}^r_i$: work in process of part type r in cell i at the end of period t_0 hence available at the beginning of period t_1.

$U^r_{i,m}(t_k)$: The expected time needed by machine m at cell i to produce *one* part of type r during time period t_k

$\eta_{i,m}(t_k)$: Efficiency (maximum allowed utilization) of machine m in cell i during time period t_k

$\mathbf{1}\{\cdot\}$: Indicator function, equal to 1 if $\{\cdot\}$ is true and 0 otherwise.

The lead time constraints that arise for each cell are generally non-linear and difficult to represent since only point estimates are usually possible. Their modeling is discussed next. Safety stock constraints arising for the horizontal cell coordination policies of Section 3 are not modeled here for simplicity. These are also non-linear constraints and can be generally modeled using a similar approach to that employed here in modeling the non-linear lead time constraints.

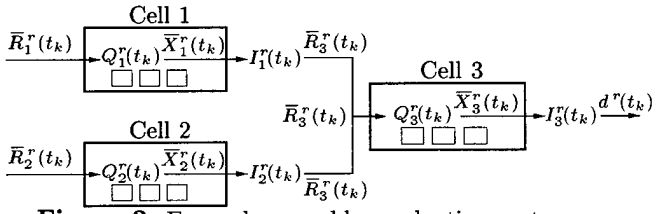


Figure 2: Example assembly production system

2.2 Non-Linear Constraint Modeling

Assuming that the duration of the time period is long enough to achieve steady state, we can use Little's law to simplify lead time constraints (lt) so that they can be written by referring to a single time period. Indeed we can write them as:

$$\overline{X}^r_i(t_k) \;\leq\; \frac{\overline{Q}^r_i(t_k)}{LT_i{}^r(t_k)}$$

Noting further that $LT_i{}^r(t_k)$ is a function of production requirements for all part types, the lead time constraint can be written as:

$$\overline{X}^r_i(t_k) \;\leq\; g^r_i\left(\overline{Q}^r_i(t_k), \overline{X}^r_i(t_k), \overline{X}^{r^*}_i(t_k)\;\forall r^* \neq r\right)$$

where g is a nonlinear function. Henceforth we will refer to the g function as the *Lead Time Constraint* function. The following representations are possible:

Lead Time Fixed at its Maximal Value: This is the approach of state of the art MRP systems. Usually, a maximal capacity utilization of cell resources is allowed, say η^{max}_i and used in the capacity constraints. the lead time corresponding to this maximal utilization is then calculated, $LT^r_{i,max}(t_k)$, and the lead time constraint is posed as $\overline{X}^r_i(t_k) \;\leq\; \overline{Q}^r_i(t_k)/LT^r_{i,max}(t_k)$. Since $LT^r_{i,max}(t_k)$ is a constant, the lead time constraint becomes a simple linear constraint which reduces the planning problem to a simple linear programming problem that requires little information exchange with the supply chain components.

The Lead Time Constraint is Modeled as a Function of Time Period Specific Cell Load: The actual surface properties of the lead time constraint functions are important here. Concavity of the lead time constraint function results in a convex feasible set which can be tractably approximated locally at any desired accuracy level through strategically selected outer-linearizations. Indeed, linearizations can be estimated by decentralized calculations of derivative values of the g function at appropriately selected points generated by the iterative Genaralized Benders' Master problem algorithm described in Section 2.3. In the absence of concavity, outer-linearization is no more effective, and constraint gradient type algorithms are resorted to. For a more detailed description of nonconvexity see [9].

2.3 Master-Problem Algorithms

The non-linear lead time constraints are handled by an iterative algorithm that mirrors the information exchange interactions between the master problem coordinator and the sub-problem decentralized decision and information feedback generation occurring at each cell.

–At iteration n, the master problem generates a tentative set of production targets which are conveyed to each cell sub-problem.

–Each cell i sub-problem examines whether production targets assigned to it are feasible, assuming that all other cells can meet their targets and therefore supply cell i with the associated input as specified by the flow equations. It then converts the tentative production targets for each period t_k in the planning horizon to a point on the lead time constraint surface. This is done by adjusting either the work in process, $Q^r_i(t_k)$, or its production requirements, $\overline{X}^r_i(t_k)$, as specified by the master problem iteration. The subproblem estimates partial derivative (or sensitivities) evaluated at that point on the surface, and returns to the master problem a linearized constraint of the form:

$$\overline{X}^r_i(t_k) \;\leq\; \widetilde{g} + \frac{\partial g}{\partial Q^r_i(t_k)}\left(\overline{Q}^r_i(t_k) - \widetilde{Q}^{r,n}_i(t_k)\right) + \sum_{r^* \neq r} \frac{\partial g}{\partial \overline{X}^{r^*}_i(t_k)}\left(\overline{X}^{r^*}_i(t_k) - \widetilde{X}^{r^*,n}_i(t_k)\right)$$

where $\left[\widetilde{Q}^{r,n}_i(t_k),\; \widetilde{X}^{r,n}_i(t_k),\; \widetilde{X}^{r^*,n}_i(t_k)\;\forall r^* \neq r\right]$ is a feasible point on the surface of the $g^r_i(t_k)$ function obtained using the tentative solution of iteration n as a starting point. In this constraint, $\widetilde{g}$, $\partial g/\partial \overline{Q}^r_i(t_k)$, and $\partial g/\partial \overline{X}^{r^*}_i(t_k)$ are the value of the function and the the partial derivatives evaluated at the point $\left[\widetilde{Q}^{r,n}_i(t_k),\; \widetilde{X}^{r,n}_i(t_k),\; \widetilde{X}^{r^*,n}_i(t_k)\;\forall r^* \neq r\right]$.

Notice that cell subproblems evaluate the information needed for the linearization using a finer time scale stochastic dynamic model. Sensitivity estimates are obtained at each sub problem and are time averaged so that they are consistent with the time scale used by the master problem.

As the iterations proceed, the master problem learns more about the non-linear lead time constraints and eventually generates the desired optimal and consistent production plan.

Concave Lead Time Constraint Function: When lead time constraint functions are concave, the sub-

problem linearization task generates a hyper plane tangent to the constraint boundary that constitutes a supporting plane for the feasible region of the lead time constraint. All linearizations, past and current, are added and retained as linear inequalities and each master problem solution is simply the solution of a linear program. The number of constraints of the Linear Program master problem increases with the progress of iterations. With the accumulation of constraints, the local representation accuracy of the non-linear lead time constraints increases arbitrarily till a convergence criterion is met. This procedure is known as the Generalized Benders' decomposition algorithm [10].

Non-concave Lead Time Constraint Function: When lead time constraint functions are not concave, the master problem is converted to a modified Generalized Benders Algorithm consisting of two phases:

–*Phase 1* accumulates all constraints generated at each iteration and the algorithm proceeds until a feasible solution is obtained. Since at least some lead time constraint functions are not concave, the linear polyhedron corresponding to the tangent hyper planes will exclude a subset of the feasible region. The successive LP master problems will converge to a solution, but the solution will not necessarily be a local optimum.

–*Phase 2* aims at proceeding to a solution that is at least locally optimal. This is achieved by discarding hyper planes generated in earlier iterations, keeping only a subset of constraints generated during a fixed number of the most recent iterations. Convergence indicates that the problem is locally convex and the solution is locally optimal. A constrained gradient method used following the convergence of phase 2 can not improve the cost of the solution verifying that the solution obtained is indeed a local optimum.

Since the phase 1-phase 2 algorithm can only guarantee a local optimum when some lead time constraint functions are not concave, the trajectory of points on the lead time constraint surface where hyper planes are generated is important as it can lead to a different local optimum. Recall that LP generated tentative production targets are generally infeasible – i.e. they violate the true lead time constraint. These targets must be therefore mapped to a point *on* the surface of the g function and a new tangent hyper plane generated at that point. Two alternative associations are used: (*i*) the *target* x or *tx* association where the tentative X value is retained unchanged while the $\overline{Q}$ value is adjusted as needed to create a point on the surface of the g function, and (*ii*) the *target* q or *tq* association where the tentative Q value is retained unchanged while the $\overline{X}$ value is adjusted.

The performance of the two phase algorithm under the *target* x and *target* q options is examined in the next section.

2.4 Computational Results

Value of Information: We briefly report results comparing a limited information exchange MRP-type algorithm to the full information exchange supply chain coordination algorithm proposed here. Three cases are reported on a 40 time period planning horizon for the three cell-three part type system of Section 2.1. We modeled cells as Closed Queueing Networks with exponentially distributed machine processing times. Demand requires an average capacity utilization of 60%, 80% and 89.3% across cases and a maximum allowed capacity utilization constraint of 90% is imposed on both the limited and full information exchange algorithms (see [5] for details). The limited information exchange algorithm uses a fixed lead time constraint which corresponds to the maximum (90%) allowed capacity utilization, and is hence least disadvantaged in case 3. Nevertheless, the proposed full information exchange algorithm achieves a 25% cost reduction even in case 3, while in cases 2 and 1 the reduction is of the order of 70% and 80%. We attribute this to the value of information on the variability of lead times. In the reasonable assumption of mild demand tracking requiring capacity utilization variations in the range of 80% to 90%, the value of information is in the vicinity of 50%.

WIP Constraint Representation	Total Cost		
	Case 1	Case 2	Case 3
Limited Information Exchange System	33,844	48,193	98,638
Full Information Exchange System	5,919	15,742	72,847

Table 2: Value of information

Effectiveness of Master Problem Algorithms, Discussion of Convergence Behavior: A simple two identical cell supply chain where each cell processes two part types on two workstations with FCFS queue protocol exponential processing times is used for simplicity and brevity. Numerical experience on larger systems such as the one used in Section 2.4 is qualitatively comparable. Three lead time constraint function types are used (*i*) type 1 refers to unambiguously concave g functions at all cells and for all part types, (*ii*) type 2 refers to some mildly non-concave g functions, and (*iii*) type 3 refers to some severely non-concave g functions brought about by considering FCFS queue protocols and significantly different part type processing times. Moreover, two demand scenarios were used to test algorithmic performance more effectively.

The following cost coefficients are used in all lead time constraint function types and demand scenarios:

$$w = \begin{bmatrix} 10 & 10 \\ 15 & 15 \end{bmatrix}, \; c^+ = \begin{bmatrix} 15 & 15 \\ 20 & 20 \end{bmatrix}, \; c^- = \begin{bmatrix} 0 & 0 \\ 200 & 200 \end{bmatrix}$$

as WIP holding cost, FGI holding cost, and FGI backlog cost, respectively. Columns are associated with part types and rows with cells. Table 3 shows production rate capacities of part types (denoted as pt 1 and pt 2) at workstations 1 and 2 (denoted as ws 1 and ws 2) for each of the three types of lead time constraint functions considered.

Table 4 shows the demand scenarios used. The demand for each part type is expressed as a percentage of the production capacity. Initial WIP and FGI is assumed to be zero at both cells. Both demand scenarios have the same total horizon demand. Scenario 2, however, has a higher demand during periods t_3 - t_{13}.
Figures 3 through 5 show the performance of the extended two phase algorithm for each of the lead time constraint function types. In each g function type figure, the objective function expressed as a proportion of the lowest feasible solution cost obtained is plotted against the iteration number. Four trajectories are plotted in each figure corresponding to each of the two demand scenarios and the *target* x or *target* q associ-

	Type 1		Type 2		Type 3	
	ws 1	ws 2	ws 1	ws 2	ws 1	ws 2
pt 1	5.00	8.33	6.25	6.25	100.00	100.00
pt 2	12.50	6.25	5.00	5.00	5.00	5.00

Table 3: Workstation production rate capacities

Time Period	Demand Scenario 1		Demand Scenario 2	
	pt 1	pt 2	pt 1	pt 2
t_1	60	32	0	0
t_2	20	64	0	0
t_3	60	48	60	48
t_4	40	48	40	48
t_5	60	64	60	64
t_6	40	32	40	48
t_7	40	32	40	48
t_8	20	48	20	64
t_9	40	48	80	48
t_{10}	40	48	60	80
t_{11}	60	32	80	48
t_{12}	40	16	40	16
t_{13}	60	48	60	48
t_{14}	0	0	0	0
t_{15}	0	0	0	0

Table 4: Demand scenarios

ation options described in Section 2.3. The first few iterations are omitted to allow better resolution in the vicinity of interest. Solutions that are infeasible are shown with a thin line while the thicker line indicates feasible solutions.

For the concave and mildly non-concave lead time constraint function types convergence is achieved by all association alternatives and demand scenarios. In fact for the concave type, phase 1 is sufficient. For the severely non-concave g function type, only the *target x* association alternative converges clearly, and at any case achieves a better solution relative to the *target q* alternative. Whenever *target q* association alternative converges, it does so faster than *target x* alternative. However *target x* alternative seems to be more robust and converges even under severely non-concave lead time constraints.

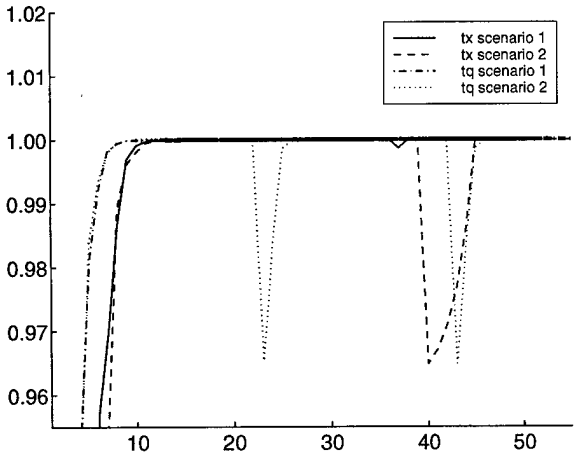


Figure 3: Concave lead time constraint function

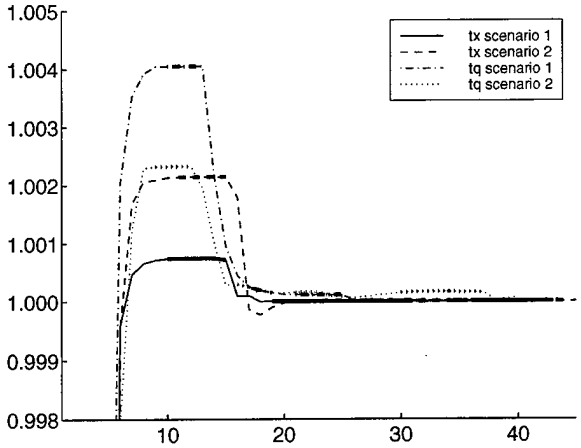


Figure 4: Mildly non-concave lead time constraint function

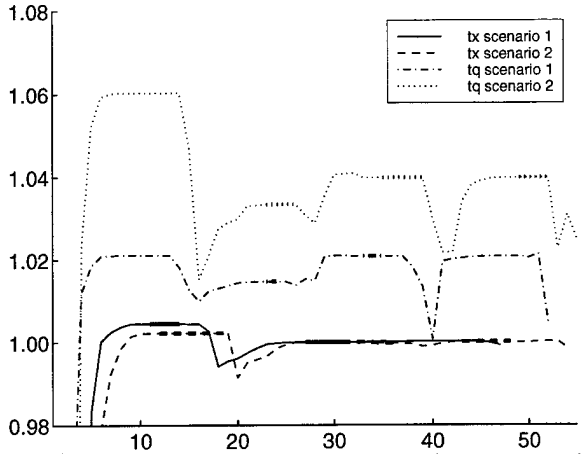


Figure 5: Severely non-concave lead time constraint function

3 Horizontal Cell Coordination

To illustrate the proposed approach let us focus on two neighboring cells that exchange multiple part types. These two cells operate as *make-to-stock* manufacturing system. In particular, we can view the requirements of the downstream cell as the *demand* and the upstream cell as the *production facility*. In between the two cells we maintain a certain level of *finished goods inventory (FGI)*. Demand is met from the FGI and the production facility strives to maintain it nonempty to avoid stockouts, which lead to backordered demand. The fundamental trade-off is between *producing*, which accumulates inventory and incurs inventory costs, and *idling*, which leads to stockouts and unsatisfied demand. In *multiclass* systems, where a single facility produces several products, an additional control action is *sequencing* or *scheduling*, that is, what product to produce, if any. We will refer to the set of rules that determine both sequencing decisions, and idling decisions, as *production policy*. The objective is to devise a production policy which optimizes some measure of the system's performance. This measure should incorporate both *inventory costs* and *backorder costs*, i.e., costs associated with not being able to meet demand at the time it arrives.

The single-class version of the problem has been stud-

ied extensively (see [11, 12] and references therein), and it has been established that a *base-stock policy* (produce when inventory falls below a certain threshold and idle otherwise) is optimal. Two-class systems have been studied in [13, 14, 15]. In the general multiclass case, [16] proposes a policy based on heavy-traffic asymptotics; other heuristics are in [17, 18]. Finally, a static allocation policy is considered in [19] for the multiclass system and asymptotically analyzed for renewal arrivals and services by essentially decomposing the system to single-class systems. In the latter paper, the author considers probabilistic service level constraints and uses asymptotics which are similar in spirit to ours but apply only under renewal assumptions (see also [20]).

The main contributions of our approach are:
1. *Probabilistic constraints to capture Quality of Service (QoS).* Most of the work in the literature considers minimizing expected *linear* inventory and backorder costs. The assumption of linear backorder costs appears to serve analytical tractability rather than an accurate representation of reality. Moreover and more importantly, it is hard to obtain data which will help quantify the consequences of unsatisfied demand via linear backorder costs. To address this need, we introduce constraints that ensure that probabilities of stockout for different products stay bounded below given desirable levels. This results in a more natural representation of the level of service provided to the downstream cell. Thus, we formulate the performance objective as: *minimize expected inventory costs subject to stockout probabilities being bounded above by given constants.*
2. *Dependencies in demand and service processes.* In practice, demand might have strong correlations with a variety of phenomena such as: the operational capacity of the downstream cell, weather patterns, state of the economy, etc. In addition, manufacturing facilities are *stochastic* and *failure-prone*, which creates dependencies in production processes. To that end, our analysis will allow demand and production to be modeled by *autocorrelated* stochastic processes. We will demonstrate that such *distributional information* (vs. knowledge of the first two moments only) is critical in optimizing performance.
3. *Fluid and large deviations techniques.* On the methodological side, we combine recently developed techniques in *fluid models* and *large deviations.*

3.1 The model

We consider a *multiclass make-to-stock manufacturing system* producing m products to be stocked *separately* in the *finished goods inventory (FGI).* Demand is met from the available FGI, and it is backordered if inventory is not available. We assume a periodic review policy where time is divided into time slots of equal duration and the system is examined at the beginning of each time slot. We let D_i^j denote the amount of class j orders arriving during time slot i, for i in the set of integers $\mathbb{Z}$ and $j = 1, \ldots, m$. We will be measuring D_i^j in production time units when the facility is working at a production rate of 1. That is, D_i^j is equal to the time that the production facility requires to produce, at a production rate of 1, the amount of class j orders arriving during time slot i. We let also B_i denote the amount of work, measured in the same production time units, that the production facility can complete during time slot i. Let finally x_i^j denote the class j inventory, measured in the same production time units, which is available at the beginning of time slot i. We allow the inventory to take negative values to denote backordering; when x_i^j is negative $-x_i^j$ is equal to the amount of work backordered from class j. We will be using the notation $\mathbf{x}_i = (x_i^1, \ldots, x_i^m)$. All the demand processes $\{D_i^j;\ i \in \mathbb{Z}, j = 1, \ldots, m\}$ and the production process $\{B_i;\ i \in \mathbb{Z}\}$ are arbitrary stationary stochastic processes that satisfy certain mild technical conditions. These conditions are satisfied by renewal processes, Markov-modulated processes, and in general stationary processes with mild mixing conditions (for details see [21, 22]). For stability purposes we assume that $\sum_{j=1}^m \mathbf{E}[D_1^j] < \mathbf{E}[B_1]$, which by stationarity carries over to all time slots i.

As discussed above, our objective is to devise a production policy that minimizes finished goods inventory costs and guarantees that the steady-state stockout probabilities $\mathbf{P}[x_i^j \leq 0]$ do not exceed some desired small values ϵ_j, for each class j. We will refer to these latter constraints as *service level* constraints.

3.2 Outline of our approach

Due to space limitations we will only present an outline of of our approach. The interested reader is referred to [12] for the detailed analysis.

We first ignore stochasticities and consider a deterministic version of the problem for which we will obtain an optimal sequencing policy. For the purposes of this deterministic version we will assume that the backlog at time slot i incurs cost at a rate of $\sum_{j=1}^m f_j(x_i^j)$ per timeslot, where the cost function $f_j(x_i^j)$ can have one of two forms: *linear* or *quadratic* (see [12]). We obtain optimal sequencing policies that minimize $\sum_{i=1}^T \sum_{j=1}^m f_j(x_i^j)$, under both cost assumptions, where T is the time horizon of interest. To that end, we introduce a continuous-time fluid model (i.e., the continuous-time analog of the deterministic version of the problem) and use calculus of variations techniques.

The fluid model evolves in continuous time and ignores stochasticities in the demand and service processes which lead to stockouts. To accommodate for these effects we consider a deterministic analog of the optimal "fluid" sequencing policies and enhance them with an appropriate idling policy. We use large deviations theory to analyze the resulting production policies and tune the parameters that characterize the corresponding idling policy. We will provide evidence (analytical and numerical) that the large deviations asymptotics are *relevant*, i.e., the analytically calculated parameters of the proposed production policies are very close to simulated values.

3.3 The proposed production policy

The fluid model mentioned above does not provide any information on the idling policy, since stochasticities are completely ignored. To that end, we will enhance the "fluid" policy to *hedge* against stochasticity and focus on a *base-stock* class of policies. In particular, we will consider the following idling policy that utilizes a so called *hedging point* or *safety stock* $\mathbf{w} = (w_1, \ldots, w_m)$:

- idle during time slot i when $\mathbf{x}_i \geq \mathbf{w}$, and
- work on the classes j that satisfy $x_i^j < w_j$, without

exceeding the corresponding safety stock w_j.

In the latter case, one of the sequencing policies determined by the fluid policy will be followed, that is,
Priority-based Policy: We define a fixed ordering $(\chi(1), \ldots, \chi(m))$ of the set of classes $\{1, \ldots, m\}$ and we work on the class j which has the highest rank $\chi(j)$ and satisfies $x_i^j < w_j$.
Generalized Longest Queue First-based Policy (GLQF): We define scalars $c_1, \ldots, c_m$ and we work on the class j that maximizes $c_j(w_j - x_i^j)$, assuming that there exists at least one j that satisfies $x_i^j < w_j$.

We are interested in determining a hedging point $\mathbf{w}$ which guarantees that the steady-state stockout probabilities $\mathbf{P}[x_i^j \leq 0]$ do not exceed some desired small values ϵ_j, for each class j.

3.4 An analytic expression for the hedging point

The exact calculation of the stockout probabilities is intractable, especially in view of the need for sophisticated (autocorrelated) models for the demand and production process. Instead, we will resort to large deviations asymptotics (see [23, 21, 22]).

In particular, the stockouts probability for class $j = 1, 2, \ldots, m$ under both the priority and the GLQF policy can be approximated by an exponential, i.e.,

$$\mathbf{P}[x^j \leq 0] \sim e^{-w_j \theta_j^*}, \qquad (1)$$

where θ_j^* is the *asymptotic decay rate* of the stockout probability and depends on the policy. We will denote it by $\theta_{P_j}^*$ in the priority case, and by $\theta_{GLQF_j}^*$ in the GLQF case. These decay rates can be obtained as the optimal value of a corresponding nonlinear programming problem that involves the limiting log-moment generating functions of the demand and production processes (see [12] for the detailed analysis). These approximations are *asymptotically exact*, that is the logarithm of the stockout probability divided by w_j converges to $-\theta_{P_j}^*$ for the priority case (respectively, $-\theta_{GLQF_j}^*$ for the GLQF case) as $w_j \to \infty$.

The asymptotics above can be further refined and used to analytically determine the appropriate hedging point $\mathbf{w}$ such that $\mathbf{P}[x_i^j \leq 0] \leq \epsilon_j$. More specifically, we will be estimating the stockout probability of class j as

$$\mathbf{P}[x^j \leq 0] \approx \alpha_j e^{-w_j \theta_j^*}, \qquad (2)$$

where θ_j^* is the appropriate decay rate depending on the production policy (either $\theta_{P_j}^*$ or $\theta_{GLQF_j}^*$). Thus, the hedging point satisfies

$$w_j = -\frac{\log(\epsilon_j/\alpha_j)}{\theta_j^*}, \qquad j = 1, \ldots, m. \qquad (3)$$

If $\theta_j^* = \infty$, then stockouts for class j do not occur and $w_j = 0$ should be used (*Just In Time (JIT)* policy). Estimates of the constant α_j are obtained in [12].

It is instructive, and indicative of the relevance of the large deviations asymptotics, to specialize our results in the single class M/M/1 case, where there are only idling decisions to be made. It can be shown (see [12]) that in this case our (large deviations asymptotics) yield the optimal idling policy.

3.5 Selecting the "best" priority policy

We now have all the ingredients to implement the proposed production policies. In the GLQF case, the costs coefficients c_j are assumed to be given and reflect inventory cost considerations. In the priority case, however, the fluid analysis determined that priority classes are ordered according to the backorder cost coefficients. We argued in the beginning of this section that these coefficients can not be assumed as given. Thus, to completely characterize the proposed production policies we are left with selecting the priority ordering. We will do that with the objective of minimizing the expected inventory cost $\sum_{j=1}^m h_j \mathbf{E}[(x^j)^+]$.

To that end, in [12] we have derived the following approximation

$$\sum_{j=1}^m h_j \mathbf{E}[(x^j)^+] \approx \sum_{j=1}^m h_j \left(w_j - \mathbf{E}[L^j] + \mathbf{E}[L^j] e^{-w_j \theta_{P_j}^*} \right),$$

where L^j is the queue length process in a corresponding make-to-order system (see [12]). We will select the priority ordering that minimizes the right hand side of the above. In practice, m is a fairly small number which makes the computation feasible. Furthermore, there are ways to make the search efficient (see [12]).

3.6 Numerical Results

We will present a 3-class example with priority scheduling. Numerical examples for the GLQF policy are in [12].

Demand and service processes are *deterministic Markov-modulated* processes (depicted in Figure 6). That is, D_i^j is a deterministic function of an underlying discrete-time Markov chain which makes one transition at each time slot. Let $\mathbf{r}_X$ denote the vector of demand or production amounts at each state of the corresponding Markov chain. In this example: $\mathbf{r}_{D^1} = (14, 3)$, $\mathbf{E}[D^1] = 9.11$, $\mathbf{Var}[D^1] = 29.88$, $\mathbf{r}_{D^2} = (2, 8, 0.5)$, $\mathbf{E}[D^2] = 2.69$, $\mathbf{Var}[D^2] = 8.87$, $\mathbf{r}_{D^3} = (5, 0.5)$, $\mathbf{E}[D^3] = 2.94$, $\mathbf{Var}[D^3] = 5.03$, $\mathbf{r}_B = (0, 20)$, $\mathbf{E}[B] = 15.56$, and $\mathbf{Var}[B] = 69.14$.

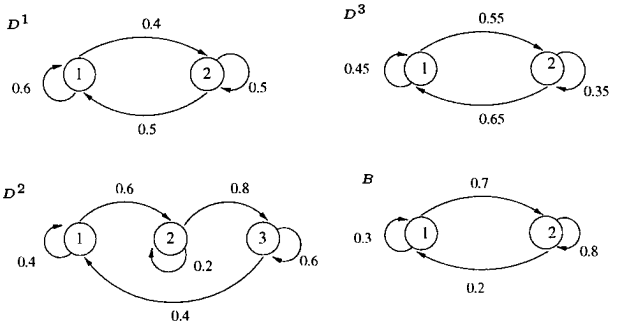


Figure 6: Demand and service processes in the three-class example with priority scheduling.

We first want to examine the accuracy of the procedure we proposed in Section 3.5 to select the best priority ordering. Table 5 presents some numerical results. In all cases reported, the optimal ordering χ given in the

	Inventory costs ($\mathbf{h}$)	Service levels (ϵ)	Optimal ordering (χ)	Hedging Point ($\mathbf{w}$)
Case 1	$(1,2,3)$	$(0.05,0.01,0.005)$	$(3,2,1)$	$(3.89,10.77,231.72)$
Case 2	$(3,2,1)$	$(0.005,0.01,0.05)$	$(1,2,3)$	$(44.37,64.58,214.26)$
Case 3	$(1,1,1)$	$(0.001,0.01,0.05)$	$(1,2,3)$	$(88.58,64.58,214.26)$
Case 4	$(3,2,1)$	$(0.01,0.01,0.01)$	$(2,1,3)$	$(4.8,73.49,349.39)$
Case 5	$(1,1,1)$	$(0.007,0.01,0.05)$	$(2,1,3)$	$(4.8,80.43,214.26)$

Table 5: We use $\mathbf{h} = (h_1, h_2, h_3)$ to denote the vector of inventory costs for the products $1, 2, 3$, respectively, $\epsilon = (\epsilon_1, \epsilon_2, \epsilon_3)$ are the desirable service levels, $\chi = (\chi(1), \chi(2), \chi(3))$ denotes the optimal priority ordering as derived by the procedure in Section 3.5, and $\mathbf{w} = (w_{\chi(1)}, w_{\chi(2)}, w_{\chi(3)})$ is the hedging point required to maintain the service levels (given by Eq. (3)).

table is identical to the one obtained by simulation. This suggests that the procedure in Section 3.5 predicts the optimal priority ordering *accurately*.

In Cases 1, 2 and 3 (notice that Cases 1 and 2 are symmetrical) the result for the optimal ordering is intuitively obvious: higher inventory cost h_j and smaller ϵ_j results in higher priority for class j. The situation in Cases 4 and 5 is more subtle. In Case 4, although all classes have the same ϵ's and class 1 has higher inventory cost than class 2, it is optimal to give top priority to class 2. The reason is that class 2 demand arrives in smaller quantities (mean 2.69 with a peak of 8 versus a mean of 9.11 and a peak of 14), which the server can handle "almost immediately", thus, when given top priority it requires very small safety stock. In addition, since it consumes only a small fraction of the capacity, serving it with top priority does not substantially affect the required safety stock for class 1 $((w_{\chi(1)}, w_{\chi(2)}) = (w_2, w_1) = (4.8, 73.49)$ for classes 2 and 1, respectively). If on the other hand top priority is given to class 1, the required safety stocks are $(w_1, w_2) = (36.53, 64.58)$ for classes 1 and 2, respectively, which justifies that it is preferable to serve class 2 with higher priority. Similarly, in Case 5 it is optimal to give top priority to class 2, although inventory costs are identical across classes and class 1 has smaller ϵ than class 2. Of course, when we bring the ϵ of class 1 sufficiently down (to 0.001) it becomes optimal to serve it with top priority (Case 3). But for this to happen the difference in the ϵ's between the two classes has to be substantial (0.001 vs. 0.01). The important conclusion is that the optimal priority ordering may depend on subtle *distributional* differences between classes that can not be captured by just the first few moments. It is interesting that these differences are captured by our method, since the large deviations behaviour depends on the whole distribution through the limiting log-moment generating functions.

We next turn our attention to the accuracy of the analytically estimated hedging point (Eq. (3)). Table 6 compares the hedging point calculated by Eq. (3) (denoted by $\mathbf{w}$ in the table) with the one obtained by simulation (denoted by $\mathbf{w}'$ in the table) for a range of service levels ϵ. Notice that to find the required hedging point via simulation, one needs to simulate the stockout probabilities for every possible value of the hedging point, which is a very computationally intensive task. The advantage of an analytically obtained hedging point is apparent. The error we report in the table is defined as $\delta_j \triangleq \frac{|w_j - w'_j|}{w'_j} \times 100\%$, for class j. We observe that the analytically obtained hedging point is fairly accurate with the error being mostly less than 3%, with the exception of two cases where the error is 3.8% and 7.52%, respectively. Recall that we measure everything in production time units, thus, the hedging point is a real number. In the simulation, however, since it is impossible to simulate for every possible value of the hedging point, we considered only integer values. As a result, the reported error includes this "quantization error", and hence underestimates the accuracy of the calculated hedging point (this "quantization" error is up to 1 time unit, that is 3.7% and 2% for the cases where we report errors of 7.52% and 3.8%, respectively).

4 Conclusions

A framework for synergistic and decentralized decision making in complex stochastic systems was described and applied to manufacturing supply chain coordination. Both vertical and horizontal coordination applications were described and algorithmic solutions proposed and tested numerically. Vertical coordination is essential for medium term decisions such as overtime labor assignment and shift management to be coordinated and optimized in decentralized decisions. Horizontal coordination is essential for short term decisions such as whether to idle or produce and what to produce to avoid short term starvation of interacting cells.

In the vertical coordination application we demonstrated that systematic distributed decision making coupled with tight coordination is capable of achieving supply chain coordination efficiencies that are superior to those obtained with state of the art methods. Furthermore we demonstrated the existence of algorithms which can coordinate supply chains effectively under complicating factors such as non-concavity of non-linear lead time constraints.

In the horizontal coordination application, we have combined fluid and large deviations techniques to derive production policies for multiclass make-to-stock manufacturing systems under realistic modeling assumptions. Our analysis ensures that the stockout probability for each product stays bounded below a desirable threshold. This leads to manufacturing systems with *Quality of Service guarantees*, a feature which we view as being increasingly important in today's competitive and service oriented environment. To provide such guarantees we require detailed *distributional* information on the stochastic processes involved. We demonstrated through numerical results that such information is critical in optimizing performance. Ignoring it, can lead to substantial performance loss. The spread of information technology in manufacturing plants and the capabilities in data collection and in the implementation of sophisticated production policies that it provides, enhance the practical significance

	Calculated $\mathbf{w}$			Simulated $\mathbf{w}'$			Error (δ)		
ϵ	w_1	w_2	w_3	w_1'	w_2'	w_3'	δ_1	δ_2	δ_3
0.1	17.14	35	181.8	17	36	181	0.82%	2.78%	0.44%
0.05	24.97	48.10	240	27	50	239	7.52%	3.8%	0.42%
0.01	43.16	78.53	375.1	44	81	372	1.91%	3.05%	0.83%
$5 \cdot 10^{-3}$	50.99	91.63	432.9	52	94	430	1.94%	2.52%	0.67%
10^{-3}	69.18	122.05	567.6	70	125	563	1.17%	2.36%	0.82%
$5 \cdot 10^{-4}$	77.02	135.15	625.6	78	139	621	1.26%	2.77%	0.74%
10^{-4}	95.20	165.58	760.2	96	169	759	0.83%	2.02%	0.16%

Table 6: Comparing analytically calculated vs. simulated hedging points. The priority ordering is fixed to (1,2,3).

of our techniques.

References

[1] V. R. Saksena, J. O'Reilly, and P. V. Kokotovic, "Singular perturbations and time scale methods in control theory: Survey 1976-1983," *Automatica*, vol. 20, no. 3, May 1984.

[2] S. B. Gershwin, *Manufacturing Systems Engineering*, Prentice Hall, 1994.

[3] C. Kaskavelis and M. Caramanis, "Integration of the production planning and control decision process in a manufacturing enterprise," in *Proceedings of the 1996 Japan-U.S.A. Symposium on Flexible Automation*, 1996.

[4] S. P. Sethi and Q. Zhang, *Hierarchical Decision Making in Stochastic Manufacturing Systems*, Birkhauser, 1994.

[5] M. C. Caramanis and O. M. Anlı, "Dynamic lead time modeling for JIT production planning," in *Proceedings of the IEEE Robotics & Automation Conference, Detroit, MI*, May 10-15, 1999.

[6] R. St. John, "The cost of inflated planned lead times in MRP systems," *Journal of Operations Management*, vol. 5, 1985.

[7] B. Millard, "JIT vs. MRP unmasking the great push-pull myth," *APICS The Performance Advantage*, vol. 8, no. 3, March 1998.

[8] H. L. Lee, "Design for supply chain management: Methods and examples," *Perspectives in Operations Management*, 1993.

[9] M. C. Caramanis and O. M. Anlı, "Modeling Load-Dependent Lead Time Constraints for Efficient Supply Chain Coordination: Convexity Issues," to appear in Proceedings of the IEEE Conference on Decision and Control, Phoenix, AZ, December 7-10, 1999.

[10] J. Bloom, M. Caramanis, and L. Charny, "Long-range generation planning using Generalized Benders' Decomposition: Implementation and experience," *Operations Research*, vol. 32, no. 2, pp. 290–312, 1984.

[11] R. Kapuscinski and S.R. Tayur, "Optimal policies and simulation based optimization for capacitated production inventory systems," in *Quantitative Models for Supply Chain Management*, S.R. Tayur, R. Ganeshan, and M. Magazine, Eds., chapter 2, pp. 7–40. Kluwer, 1999.

[12] D. Bertsimas and I. Ch. Paschalidis, "Probabilistic service level guarantees in make-to-stock manufacturing systems," to appear *Operations Research*, available at `http://ionia.bu.edu`, 1999.

[13] Y.-S. Zheng and P. Zipkin, "A queueing model to analyze the value of centralized inventory information," vol. 38, no. 2, pp. 296–307, 1990.

[14] A.Y. Ha, "Optimal dynamic scheduling policy for a make-to-stock production system," vol. 45, no. 1, pp. 42–53, 1997.

[15] F. de Véricourt, F. Karaesmen, and Y. Dallery, "Dynamic scheduling in a make-to-stock system: A partial characterization of optimal policies," Preprint, 1998, to appear *Operations Research.*

[16] L. M. Wein, "Dynamic scheduling of a multiclass make-to-stock queue," *Operations Research*, vol. 40, pp. 724 – 735, 1992.

[17] A. Peña Perez and P. Zipkin, "Dynamic scheduling rules for a multiproduct make-to-stock queue," vol. 45, no. 6, pp. 919–930, 1997.

[18] M.H. Veatch and L.M. Wein, "Scheduling a make-to-stock queue: Index policies and hedging points," vol. 44, no. 4, pp. 634–647, 1996.

[19] P. Glasserman, "Allocating production capacity among multiple products," vol. 44, no. 5, pp. 724–734, 1996.

[20] P. Glasserman, "Bounds and asymptotics for planning critical safety stocks," vol. 45, no. 2, pp. 244–257, 1997.

[21] D. Bertsimas, I. Ch. Paschalidis, and J. N. Tsitsiklis, "Asymptotic buffer overflow probabilities in multiclass multiplexers: An optimal control approach," vol. 43, no. 3, pp. 315–335, 1998.

[22] D. Bertsimas, I. Ch. Paschalidis, and J. N. Tsitsiklis, "Large deviations analysis of the generalized processor sharing policy," *Queueing Systems*, in press, available at `http://ionia.bu.edu`, 1997; Revised 1999.

[23] D. Bertsimas, I. Ch. Paschalidis, and J. N. Tsitsiklis, "On the large deviations behaviour of acyclic networks of G/G/1 queues," *The Annals of Applied Probability*, vol. 8, no. 4, pp. 1027–1069, 1998.

Mathematical Programming Approaches for Optimization and Control of Hybrid Systems

Vipin Gopal
Honeywell Technology Center
3660 Technology Drive, Minneapolis, MN 55112
vipin@htc.honeywell.com

Abstract

Characterization of real-world systems are most often hybrid, with interacting components of continuous and discrete elements. This paper will provide an overview of the advances made in the optimization and control of hybrid systems via mathematical programming techniques. Examples are drawn from process engineering applications.

1 Introduction

Are process operations purely continuous or purely discrete? Very often the answer is neither! Characterization of dynamic physicochemical problems has both continuous and discrete components to it, that they can be termed as hybrid systems in general. Until recently, however, engineers and scientists have mostly concentrated their efforts on formulating and solving problems that are pure in a mathematical sense - they rarely had discrete and continuous components in it at the same time. This was based on the realization that models were formulated for a certain region of applicability and they perform well when applied within that range. However, their validity in the regions beyond the ones they were derived for, continues to be a question mark. Influenced by such practices, evolution of modeling and solution technology has followed a very characteristic pattern - operations are described by discrete models each of which are continuous in nature. The set of discrete models are in fact a conditional model, an if-then-else loop will explicate the conditions under which each element of this set becomes a valid model. This modeling paradigm assumes that the conditions for determining the correct model are known in advance, which is often untrue as the evaluation of the conditions may often be dependent on the solution of the model itself.

Where does automation technology stand in the wake of all this? Given a particular continuous model, technology for optimization and control using such models is maturing fast, the key assumption being that the operation stays within the prescribed bounds of applicability of the model. A big step forward would be to answer the question: can one perform similar functions using a conditional model - a hybrid system which has both discrete and continuous components to it?

Significant activity has been taking place in the hybrid systems community to address these issues. Researchers in mathematical programming, in particular, have made interesting and promising advances during the past 3-4 years in the quest for systematic solution approaches for the problems in this class. Contributions come from a broad spectra of academic disciplines - including operations research, applied mathematics, computer science, chemical engineering and civil engineering.

The initial focus of the work in this area has been in addressing *steady-state* problems, for example, design optimization and determi-

nation of optimal operating point. However, very recently, motivated by the progress made in the steady-state domain, researchers have started looking at *dynamic* hybrid systems as well. Mathematical programming approaches to dynamic hybrid systems is still far from being mature - albeit holds a lot of promise. This paper will briefly review the work accompished in this area so far, with some directions for future research.

In section 2, we will describe the key aspects of the problem and in subsequent sections, the solution methodologies. Illustrations are drawn primarily from process engineering.

2 Problem Description

For purposes of illustration, some of the fundamental characteristics of a hybrid system optimization problem are described below[1, 20]. They include:

- Distinct regions (r) in which different models are enforced. These could result in disjunctive formulations in the objective function and/or in the constraint set.

- Variables - differential $x^{(r)}(t)$ and algebraic $y^{(r)}(t)$. Time dependent inputs $u(t)$ and time invariant parameters p.

- Sets of equalities (and inequalities) in each region

$$\begin{aligned} f^{(r)}(x, x', y, u, p, t) &= 0 \\ h^{(r)}(x, x', y, u, p, t) &\leq 0 \end{aligned} \qquad (1)$$

- Point constraints valid at specific points in time t_q. These could include the initial and end points.

$$\begin{aligned} g^{(r)}(x, x', y, u, p, t_q) &= 0 \\ p^{(r)}(x, x', y, u, p, t_q) &\leq 0 \end{aligned} \qquad (2)$$

- Set of conditions for the system to be in region (r), and/or set of transitions from r' to r, $T_{rr'}$.

Having described the fundamental features of the problem, from a solution standpoint, one has to, at least, address the following issues:

- *Combinatorics* - At any given time t, the system could be in any region k. If one where to look at the system at a finite number of time points, the number of states the system could possibly be in, is a combinatorial function of the total number of available regions.

- *Dynamics* - The constraints are typically differential and algebraic in nature, ie., a *Differential Algebraic Equation (DAE)* system. The differential equations could be ordinary or partial. One can also conjecture cases where integral constraints also play a role.

- *Nonlinearity* - Constraints (1) and (2) could, in general, be nonlinear. Constraints like the ones arising in linear control applications could be considered as special cases.

3 Simulation and Modeling

Mathematical programming approaches for the optimization and control of hybrid systems have benefited heavily from the understanding gained from the study of modeling and simulation for such systems. Simulation and modeling of hybrid systems is not the focus of this paper - the reader is refered to [2, 4, 27, 39, 43] for the description of such work. Some specific areas within the realm of this topic that have had a great influence on the current research activity in optimization and control of hybrid systems include, properties of DAEs such as index [9, 22], detection and location of model changes [14, 44] and initialization and reinitialization after discontinuities of DAEs [15, 30, 42].

4 Optimization

Optimization of hybrid systems pose unique challenges and many of the recent approaches for solving this problem have attempted to bring together the relevant concepts from many disciplines. In fact, one could view this research as a confluence of research in areas such as combinatorial optimization, differential algebraic equations, numerical analysis and logic.

In particular, most of the approaches proposed for optimization of hybrid systems have a component incorporating popular approaches for the solution of dynamic optimization problem of continuous systems. Below, we briefly outline these methods.

4.1 Dynamic Optimization of Continuous Systems

The roots of dynamic optimization and optimal control can be traced to calculus of variations in the early 17th century (Bryson [10], Goldstine [25]). The area has evolved tremendously since then - a couple of notable events being the introduction of dynamic programming by Bellman [5] and that of the maximum principle by Pontryagin [47]. In the recent past, two classes of approaches have been dominant for the solution of these problems: *indirect* and *direct* methods.

The indirect methods rely on Pontryagin's maximum principle [47] and are based on solution of the first order necessary conditions for optimality. An equivalent two-point boundary value problem is derived and its solution is found. Any of single shooting, multiple shooting and global methods are used for the solution of the BVP (see Bryson and Ho [11] for details). The indirect method works well for unconstrained problems, but the presence of inequalities in the model poses difficulties for these methods (Jones and Finch [34], Ray [50]).

On the other hand, the direct approach relies on applying a nonlinear programming solver to a finite-dimensional model which is obtained from the transformation of the original infinite-dimensional problem through discretization. Most of the algorithms in this approach can be grouped into: *Control Parameterization Techniques* and *Simultaneous Methods.*

Control parametrization techniques [35, 41, 46, 54] use a sequential optimization procedure, where an optimization routine is used to calculate the control profiles and a differential equation solver is used to evaluate the values of the constraints and the objective function. The method has the advantage that the optimization problem is in the reduced dimension of the degrees of freedom of the problem and also makes use of powerful integrators like DASSL [45] for tackling stability issues. However, it has a disadvantage in that it is limited to problems that can be integrated stably in all possible control configurations, as it requires feasible solutions at all intermediate trial points.

In the simultaneous approach, the optimization and the solution of the differential algebraic equations (DAEs) are considered together in the NLP problem formulation [13, 17, 37, 51, 52]. Here a suitable discretization strategy (e.g, implicit Runge-Kutta) is used to approximate both the state and control profiles. This method has advantages of dealing with profile and path constraints through bounds or inequalities which are a natural part of the NLP. It also avoids any issues related to infeasibilities in the intermediate solutions. The simultaneous method provides an efficient way for solving highly constrained problems, but at the expense of solving large NLPs [28].

The direct methods have seen significant applications in the hybrid systems problem as well.

4.2 Dynamic Optimization of Hybrid Systems

Primarily, mathematical programming approaches for the Dynamic Optimization of Hybrid Systems (DOHS) have targeted two classes of problems - one, where the number,

nature and sequence of the different regions are known *a priori*; and the other, where they are not. In the former class, significant contributions include [40, 55, 56]. In the remainder of the section, we describe some representative work from the latter class.

4.2.1 Integer Programming Approaches

This class of solutions is based on a popular concept in combinatorial optimization - *Mixed Integer NonLinear Programming (MINLP)* [31]. MINLP provides an optimization framework that can accommodate both the discrete and continuous features of the problem. The integer (mostly binary) variables capture the discrete nature of the problem and the continuous variables (x, y) are represented within the nonlinear optimization subproblem in the MINLP formulation. The binary variables can either represent discrete decisions, such as choice of production facility, or act as a flag that can depict the current region of operation, or even denote a switch from one region to another. In the overall problem, the constraint set appear as disjunctions, ie., a set of constraints of which at least one must be true (Refer [3] for more on disjunctive programming). The relevant set of constraints for each region in consideration are made active, and the irrelevant ones inactive, through their reformulations using binary variables. Other than the constraints, the objective function could be hybrid as well, for example, various cost functions under different conditions [53].

The constraints of the MINLP formulation, with the discrete variables aside, are in the form of DAEs. Most of the popular approaches to the solution of (steady-state) MINLPs - Branch and Bound [8, 32], Generalized Benders Decomposition [23] and Outer Approximation [18, 19] - involve solution of Nonlinear Programming Problems (NLP) which are obtained by assigning fixed values to the discrete variables. However, in this case, we have an infinite dimensional dynamic optimization problem instead.

Avraam *et al.* [1] describes a framework for the modeling and optimization of hybrid systems. Their approach, innovative for being the first published work of its kind, is reproduced below. First, some definitions:

- Time periods $k = 1, ..., K$ the end of each is at time τ_k
- Binary variables X_{rk} which takes a value of 1 if the system is in region r in period k, and 0 otherwise.
- Binary variables $L_{rr'kn}$ which takes a value of 1 if the transition condition n (3) from r to r' is true at τ_k.

$$l^{(rr'n)}((x, x', y, u, p, t)) < 0 \quad \forall n = 1, ..., N^{rr'} \quad (3)$$

- Transition functions (4) associated with transition functions (3), which relates the values of variables in r to that in r' at the transition time τ_k

$$I^{(rr')}(x^{(r')}, x'^{(r')}, y^{(r')}, x^{(r)}, x'^{(r)}, y^{(r)}, u, p, t) \quad (4)$$

- Binary variables $L_{rr'k}$ which takes a value of 1 if transition from r to r' takes place at τ_k.
- L_{eq} and U_{eq} are lower and upper bounds of the corresponding equations.

The key aspect of their approach is the modeling formalism, which represents the DOHS problem in an MINLP framework. The corresponding model is:

$$\sum_r X_{rk} = 1 \quad \forall k \quad (5)$$

$$L_f^{(r)}(1 - X_{rk}) \leq f^{(r)}(x, x', y, u, p, t) \leq U_f^{(r)}(1 - X_{rk}) \quad \forall r, k \quad (6)$$

$$h^{(r)}(x, x', y, u, p, t) \leq U_h^{(r)}(1 - X_{rk}) \quad \forall r, k \quad (7)$$

$$L_l^{(rr'n)} L_{(rr'kn)} + \epsilon \leq l^{(rr'n)}((x, x', y, u, p, \tau_k)) \leq U_l^{(rr'n)}(1 - L_{(rr'kn)}) \quad \forall k, s, n \quad (8)$$

$$L_{rr'k} \leq \sum_n L_{rr'kn} \quad \forall s, k \quad (9)$$

$$L_{rr'k} \leq X_{rk} \quad \forall s, k \quad (10)$$

$$L_{rr'k} \geq L_{rr'kn} + X_{rk} - 1 \quad \forall s, k, n \quad (11)$$

$$X_{r'k+1} \geq L_{rr'k} \quad \forall s,k \tag{12}$$

$$X_{r'k+1} \geq X_{r'k} - \sum_{rr'} L_{rr'k} \quad \forall s,k \tag{13}$$

$$L_I^{(rr')}(1 - L_{rr'k}) \leq I^{(rr')}(x^{(r')}(\tau_k), x'^{(r')}(\tau_k), y^{(r')}(\tau_k), x^{(r)}(\tau_k), x'^{(r)}(\tau_k), y^{(r)}(\tau_k), u(\tau_k), \nu, t) \leq U_I^{(rr')}(1 - L_{rr'k}) \quad \forall r,k \tag{14}$$

$$-\int_{\tau_{k-1}}^{\tau_k} \min(0, l^{(rr'n)}(.))dt \leq M_k(1 - X_{rk}) \tag{15}$$

Bounds on variables (16)

Initial, final and point conditions (17)

(5) ensures that the system is in only one region at any given period. (6) and (7) enforce that the corresponding constraints are active if the system is in region r. (8) are the transition triggering conditions which sets the correponding binary variables to 1 if any of the conditions are true. Conditions (9-10) enforce that the value of the transition variable is zero if none of the triggering conditions are true or if the system is not is region r. (11) sets it to 1, if the conditions are met. Conditions (12-13) modifies the X's in the next time period accordingly. (14) sets the initial conditions in the next time period in the event of a transition. (15) ensures that a transition does not occur in the middle of a time period.

This is a infinite dimensional dynamic optimization problem having both discrete and continuous variables. A complete discretization approach with collocation on finite elements, originally proposed in [16], is followed to convert the original infinite dimensional dynamic optimization problem to a finite dimensional MINLP. Here, the time horizon is divided into finite elements and polynomial approximations for differential, algebraic and control are used within these elements. Equations (6,7,16) are enforced at all the collocation points. Transition conditions (8, 14) are enforced at the grid points (between periods). Continuity conditions on differential variables are enforced at each finite element. The duration of the finite elements in each period are allowed to vary.

If there are a common set of equations which are common to all the regions, one could also modify (6, 7) by enforcing those constraints in all cases and using the binary variables to choose from the remaining set of equations.

Approaches in the same class have also been developed by the author at Honeywell [26]. [20, 21] describe the formulation and solution of DOHS using an approach primarily based on sensitivity analysis. The study contrasts with an earlier work [38], which discusses the inferior properties of gradient based methods for DOHS.

4.2.2 Unified Formulations

In the previous section, we discussed approaches where individual models in their respective regions are enforced seperately, as and when they are valid. An interesting alternative to this approach is to derive a single consistent formulation that will reduce to the relevant set of equations in their respective regions of validity. Though a rigorous theoretic framework generalized to an arbitrary hybrid system is yet to appear in this area, specific applications have generated much interest. For illustrating the concepts, let us consider a class of problems - the most frequent computations encountered in process engineering - calculation of phase equilibrium.

Consider a simple isothermal flash. A feed stream of composition $\boldsymbol{z}$ enters the flash column at a specified temperature and pressure. The products are a vapor phase stream of composition $\boldsymbol{y}$ and flowrate V and/or a liquid phase stream of composition $\boldsymbol{x}$ and flowrate L. Different sets of equations are valid depending on the number of phases present at equilibrium. For example, the equilibrium equations are generally not valid in either of the single phase regions. The relevant set of equations are:
Two phase liquid-vapor:

$$\begin{aligned} &F - L - V = 0 \\ &Fz_i - Lx_i - Vy_i = 0 \qquad i = 1,n \\ &y_i - K_i(P,T,x)x_i = 0 \qquad i = 1,n \end{aligned}$$

$$\sum_i y_i - \sum_i x_i = 0 \qquad (18)$$

Single phase liquid:

$$\begin{aligned} & F = L \\ & z_i = x_i \qquad i = 1,n \end{aligned} \qquad (19)$$

Single phase vapor:

$$\begin{aligned} & F = V \\ & z_i = y_i \qquad i = 1,n \end{aligned} \qquad (20)$$

The subscript i denotes the components in the stream and Ks are the equilibrium constants. This is a hybrid system, where different sets of equations are valid under different conditions. However, one of the basic problems associated with phase equilibrium calculations, is that the number of phases is not known *a priori*. As an alternative to solving this problem using multiple models (18), (19) and (20) in a sequential if-then-else fashion, single consistent formulation mechanisms have been proposed where the relevant set of equations fall into place automatically. An optimization formulation (21), based on relaxation of the equilibrium expression is derived in [27] and a complementarity formulation (22) is proposed in [29]. Alternately, [12] suggests formulation based on pressure perturbation.

$$\begin{aligned} & Min\ \delta \\ s.t.\quad & F - L - V = 0 \\ & Fz_i - Lx_i - Vy_i = 0 \qquad i = 1,n \\ & y_i - \gamma K_i(P,T,x)x_i = 0 \qquad i = 1,n \\ & \sum_i y_i - \sum_i x_i = 0 \\ & \delta \geq \gamma - 1 \\ & \delta \geq 1 - \gamma \\ & \delta \geq 0 \\ & 0 \leq x_i, y_i \leq 1 \\ & 0 \leq L, V \leq F \end{aligned} \qquad (21)$$

$$\begin{aligned} & F - L - V = 0 \\ & Fz_i - Lx_i - Vy_i = 0 \qquad i = 1,n \\ & y_i - \gamma K_i(P,T,x)x_i = 0 \qquad i = 1,n \\ & \sum_i y_i - \sum_i x_i = 0 \\ & \gamma - 1 = s_- - s_+ \\ & s_+ L = 0 \\ & s_- V = 0 \\ & s_+, s_- \geq 0 \\ & 0 \leq x_i, y_i \leq 1 \\ & 0 \leq L, V \leq F \end{aligned} \qquad (22)$$

(21) and (22) can be applied to the two-phase as well as the single-phase cases. Both formulations have their roots in principles of thermodynamics, specifically, optimality conditions of the minimization of Gibbs free energy. Intuitively, the formulations are simple. In (22), the equilibrium expression is relaxed through the introduction of γ. γ takes a value different from 1 in the single phase regions, which forces either s_- or s_+ to be positive. It can be shown that s_- will be positive in the single phase liquid region. Accordingly, the complementarity condition $s_- V = 0$ ensures that $V = 0$. Similar arguments hold for the single phase vapor region. Formulations such as (21) and (22), where the relevant set of equations automatically fall into place for either single phase or two phase solutions, is helpful in dealing with phase transitions without additional procedures, in the larger context of equation based process and plant simulation and optimization. Note that addition of just three variables γ, s_-, s_+ gives it the additional capability of dealing with operation in the single phase regions. Similar formulations have been posed for multiphase systems also [29].

The author and co-workers used the above formulations to model a (maximum of) two-phase distillation column in steady-state as well as dynamic operating conditions [26, 29]. For a distillation column with N trays, the number of phase combinations computationally possible is 3^N, even though a large fraction of them are physically infeasible. We are looking at as many models as well, if we were to

model each tray in the column using (18), (19) and (20). On the other hand, computations for optimal start-up and shut-down operations of these columns need to take into account the possibility of single phases on trays, models for which are different from that of two phases, as described earlier. A single consistent formulation such as (21) and (22), eliminates the need to use an intractable number of models, thus making computations easier, faster and practical.

5 Control

Research in control has not been lagging behind either, in addressing problems that are hybrid in nature. In fact, control applications can benefit immensely from advances in treatment of hybrid systems. Historically, control applications in manufacturing processes have relied on linear models derived from step and impulse responses, even though the underlying behavior are fundamentally nonlinear. Such models have limited range of validity for reasons outlined before. The ability to handle many such models, each of which is valid in their respective regions, will lead to undoubtedly lead to better control.

At the heart of any optimal control based application is an optimization problem, and hence, optimal control of hybrid systems directly benefit from the advancements described in section 4. Of particular interest are Model Predictive Control (MPC) schemes, the most widely used implementation of multivariable optimal control in process industries. However, it is the particular properties and structure of these problems that make it appealing to develop specialized algorithms for their solution. Research is well under way in this direction as well.

[7] describes a methodology for the control of systems described by linear dynamic equations subject to linear inequalities involving real and integer variables. The control problem is posed and solved as a Mixed Integer Quadratic Programming (MIQP) [36] problem. The framework proposed in the paper, Mixed Logic Dynamical(MLD) systems, is used for modeling and controlling systems described by interdependent physical laws, logic rules and operating constraints. The use of logic is a crucial one - it has been shown before that the solution space provided by combinatorial systems could be huge and the introduction of logical constraints, from problem structure and practical process knowledge, could vastly simplify the solution process [33, 48, 49]. Further, [6] proposes an MIQP based solution strategy for MPC problems.

The team at Honeywell [24] have also proposed to use hybrid MPC schemes for the control of systems described by both discrete and continuous components, in the presence of uncertainty.

6 Conclusions

A significant surge in interest has been seen in the research and development activities for enabling better computations of hybrid systems, in the past 3-4 years. This trend has been motivated by many factors - the theoretical and practical challenges posed by the hybrid systems, the relative maturity of uni-modal implementations and the increased realization that the fundametally hybrid nature of real-world systems should be addressed. This paper provides an overview of the advances made in the development of approaches based on mathematical programming for the control and optimization of hybrid systems. The advancements so far have been encouraging and undoubtedly poised to make a difference in the future. Future work is likely to focus on better modeling paradigms and computationally efficient solution algorithms with a rigorous theoretical basis. The confluence of all this will, hopefully, result in a unified theorem of hybrid systems.

References

[1] M.P. Avraam, N. Shah, and C.C. Pantelides.

Modeling and optimization of general hybrid systems in the continuous time domain. *Computers Chem. Engng.*, 22(S):S221–S228, 1998.

[2] A. Back, J. Guckenheimer, and M. Myers. A dynamical simulation facility for hybrid systems. In *Hybrid Systems*, Lecture Notes in Computer Science, pages 255–267, 1993.

[3] E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM J.Alg.Disc.Meth.*, 6:466–486, 1985.

[4] P.I. Barton. *The Modeling and Simulation of Combined Discrete/Continuous Processes.* PhD thesis, Imperial College of Science, Technology and Medicine, London, UK, 1992.

[5] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, 1957.

[6] A. Bemporad, G. Ferrari-Trecate, D. Mignone, M. Morari, and F.D. Torrisi. Model predictive control-ideas for the next generation. In *European Control Conference*, Karlsruhe, Germany, 1999.

[7] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. Technical Report AUT-98-04, IFA ETH, 1998.

[8] B. Borchers and J.E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programming. *Computers and Operations Research*, 21:369–367, 1994.

[9] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* SIAM, Philadelphia, 1996.

[10] A. E. Bryson Jr. Optimal control-1950 to 1985. *IEEE Control Systems*, 16(3):26–33, 1996.

[11] A.E. Bryson Jr. and Y. Ho. *Applied Optimal Control: Optimization, Estimation and Control.* Hemisphere, 1975.

[12] L. G. Bullard and L. T. Biegler. Iterated linear programming strategies for nonsmooth simulation: a penalty based method for vapor-liquid equilibrium applications. *Computers Chem. Engng.*, 17(1):95–110, 1993.

[13] M.D. Canon, C.D. Cullum Jr, and E. Polak. *Theory of Optimal Control and Mathematical Programming.* McGraw Hill, New York, NY, 1970.

[14] M. B. Carver. Efficient integration over discontinuities in ordinary differential equations. In A.W. Bennett and R. Vichnevetsky, editors, *Numerical Methods for Differential Equations and Simulation.* North-Holland Publishing Company, 1978.

[15] Y. Chung and A.W. Westerberg. A proposed numerical method for solving nonlinear index problems. *Ind. Eng. Chem. Res.*, 29(7):1234–1239, 1990.

[16] J.E. Cuthrell and L.T. Biegler. On the optimization of differential-algebraic process systems. *AIChE J.*, 33:1257–1270, 1987.

[17] J.E. Cuthrell and L.T. Biegler. Simultaneous optimization and solution methods for batch reactor control profiles. *Computers Chem. Engng.*, 13(1/2):49–62, 1989.

[18] M. Duran and I.E. Grossmann. An outer-approximation algorithm for a class of mixed integer nonlinear programming. *Mathematical Programming*, 36:307–339, 1986.

[19] R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66(3):327–349, 1994.

[20] S. Galan and P.I. Barton. Dynamic optimization of hybrid systems. *Computers chem. Engng.*, 22(S):S183–S190, 1998.

[21] S. Galan, W.F. Feehery, and P.I. Barton. Parametric sensitivity functions for hybrid discrete/continuous systems. *Applied Numerical Mathematics*, 31:17–47, 1999.

[22] C.W. Gear. Differential-algebraic equation index transformation. *SIAM J. Sci. Stat. Comput.*, 9(1):39–47, 1988.

[23] A.M. Geoffrion. Generalized benders decomposition. *J. Opt. Theory and Appl.*, 10:237–260, 1972.

[24] D. Godbole, V. Gopal, J. Jelinek, B. Morton, D. Musliner, and T. Samad. Multi model predictive control of air operations. In *AEC Symposium*, San Diego, 1999.

[25] H. H. Goldstine. *A History of Calculus of variations from the 17th through the 19th century.* Springer-Verlag, 1980.

[26] V. Gopal. Optimization of hybrid systems. Technical report, Honeywell Technology Center, 1998.

[27] V. Gopal and L. T. Biegler. Nonsmooth dynamic simulation with linear programming based methods. *Computers Chem. Engng.*, 21(7):675–689, 1997.

[28] V. Gopal and L. T. Biegler. Large scale inequality constrained optimization and control. *IEEE Controls Systems Magazine*, 18(6):59–68, 1998.

[29] V. Gopal and L. T. Biegler. Smoothing methods for the treatment of complementarity conditions and nested discontinuities. *AIChE J.*, 45(7):1535–1547, 1999.

[30] V. Gopal and L. T. Biegler. A successive linear programming approach to consistent initialization and reinitialization after discontinuities of differential algebraic equations. *SIAM J. of Scientific Computing*, 20(2):447–467, 1999.

[31] I.E. Grossmann and Z Kravanja. Mixed-integer nonlinear programming: A survey of algorithms and applications. In L.T. Biegler et al, editor, *Large-Scale Optimization with Applications, Part II: Optimal Design and Control*, pages 73–100. Springer-Verlag, 1997.

[32] O.K. Gupta and V. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31(12):1533–1546, 1985.

[33] J.N. Hooker. Logic-based methods for optimization. In A. Boring, editor, *Principles and Practice of Constraint Programming*, volume 874 of *Lecture Notes in Computer Science*, pages 336–349, 1994.

[34] D.I. Jones and J.W. Finch. Comparison of optimization algorithms. *Int. J. Control*, 40(4):747–761, 1984.

[35] D. Kraft. On converting optimal control problems into nonlinear programming problems. *Comput. Math. Prog.*, 15:261–280, 1985.

[36] R. Lazimy. Improved algorithm for mixed-integer quadratic programs and a computational study. *Mathematical Programming*, 32:100–113, 1985.

[37] J.S. Logsdon. *Efficient determination of optimal control profiles for differential algebraic systems.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1990.

[38] J. Lu, L.-Z. Liao, A. Nerode, and J.H Taylor. Optimal control of systems with continuous and discrete states. In *32nd Conf. Decision and Control*, pages 2292–2297, San Antonio, 1993.

[39] W. Marquardt. Dynamic process simulation - recent progress and future challenges. In Y. Arkun and W.H. Ray, editors, *Chemical Process Control*, pages 131–179. CACHE-AIChE Publications, 1991.

[40] K.R. Morison and R.W.H Sargent. Optimization of multistage processes described by differential-algebraic equations. In *4th IIMAS Workshop on Numerical Analysis*, Guanajuato, Mexico, 1984.

[41] I.M. Mujtaba and S. Macchietto. Optimum operation of multicomponent batch distillation-multiperiod formulation and solution. *Computers Chem. Engng.*, 17(12):1191, 1993.

[42] C. C. Pantelides. The consistent initialization of differential - algebraic systems. *SIAM J. Sci. Stat. Comput.*, 9(2):213–231, 1988.

[43] C. C. Pantelides and P. I. Barton. Equation-oriented dynamic simulation: Current status and future perspectives. *Computers Chem. Engng.*, 17(Suppl.):S263–S285, 1993.

[44] T. Park and P. I. Barton. A new algorithm for the accurate and efficient location of state events. In *Special Topical Conference on Industrial Chemical Technology, AIChE Annual Meeting*, St. Louis, MO, 1993.

[45] L.R. Petzold. DASSL: A differential/algebraic system solver. Technical Report 82-8051, Sandia National Laboratory, 1982.

[46] G.P. Polard and R.W.H Sargent. Off-line computation of optimum control for a plate distillation column. *Automatica*, 6:59–76, 1970.

[47] L.S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The mathematical theory of optimal processes.* Interscience Publishers, New York, 1962.

[48] R. Raman and I.E. Grossmann. Relation between MINLP modeling and logical inference for chemical process synthesis. *Computers chem. Engng.*, 15(2):73–84, 1991.

[49] R. Raman and I.E. Grossmann. Modelling and computational techniques for logic based integer programming. *Computers chem. Engng.*, 18(7):563–578, 1994.

[50] W. H. Ray. *Advanced Process Control.* McGraw Hill, New York, 1981.

[51] J.G. Renfro. *Computational studies in the optimization of systems described by differential/algebraic equations.* PhD thesis, University of Houston, Houston, TX, 1986.

[52] P. Tanartkit. *Stable Computational Procedures for Dynamic Optimization in Process Engineering.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1996.

[53] M. Turkay and I.E. Grossmann. Structural flowsheet optimization with complex investment cost functions. *Computers Chem. Engng.*, 22(4/5):673–686, 1998.

[54] V. Vassiliadis. *Computational Solution of Dynamic Optimization Problems with General Differential-Algebraic Equations.* PhD thesis, University of London, London, UK, 1993.

[55] V.S. Vassiliadis, R.W.H. Sargent, and C.C Pantelides. Solution of a class of multistage dynamic optimization problems. 1. problems without path constraints. *Ind. Eng. Chem. Res.*, 33:2111–2122, 1994.

[56] V.S. Vassiliadis, R.W.H. Sargent, and C.C Pantelides. Solution of a class of multistage dynamic optimization problems. 1. problems with path constraints. *Ind. Eng. Chem. Res.*, 33:2123–2133, 1994.

Adaptive Multi-platform Scheduling in a Risky Environment*

Dimitri P. Bertsekas
Dept. of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA 02139

Michael L. Curry
ALPHATECH, Inc. 50 Mall Road Burlington, MA 01803

David A. Castañon
Dept. of Electrical and Computer Engineering, Boston University, Boston, MA 02215

David Logan
ALPHATECH, Inc. 50 Mall Road Burlington, MA 01803

Abstract

In this paper, we investigate the use of rollout algorithms for adaptive multi-platform scheduling in a risky environment. The underlying decision problem is motivated by several Air Force applications: data collection, sensor management, and air operations planning. These problems may be solved optimally with stochastic dynamic programming (SDP), but have overwhelming computational requirements. Rollout algorithms reduce computational requirements by using on-line learning and simulation to approximate SDP with a base heuristic. While they do not aspire to optimal performance, rollout algorithms typically result in a consistent and substantial improvement over the underlying heuristics. A multi-platform planning and scheduling problem is used to demonstrate rollout performance.

1 Introduction

The planning and execution of multiple missions in the presence of risk is a problem which arises in many important military contexts. In data collection applications, multiple UAV platforms may be tasked to interrogate different areas, with the risk of platform destruction as each platform pursues its collection mission. In attack air operations, multiple platforms follow risky trajectories to attack enemy targets. For both applications, sensors and communication equipment can provide up-to-date information concerning individual mission and platform status, and thus provide notification of platform losses. This creates opportunities for retasking surviving platforms in order to best achieve mission objectives.

In mathematical terms, the above class of problems can be viewed as a sequential decision problem, where each decision is based on the observation of certain discrete events. These decisions affect the evolution of a system state (mission), which is also influenced by random discrete events (e.g. platform destruction). The goal is to select the current decisions as a function of the current system state, in a manner that optimizes mission performance.

The above class of problems can be formulated as Markov decision problems [3],[5]. The principal approach for solving such problems is dynamic programming (DP), which selects feedback rules to determine optimal controls for each possible state. These optimal controls are determined by evaluating at each stage the immediate expected cost of the current decision, plus the future optimal cost-to-go over future decisions. However, it is well known that computation of the optimal cost-to-go for each future state is computationally intractable for all but the simplest of problems, making direct application of DP an impossible task for multi-platform control.

In recent years, there has been a great deal of research on approximate DP methods based on computing suitable approximations to the optimal cost-to-go. These methods are collectively known as neuro-dynamic programming (NDP) [1]. In NDP, the optimal cost-to-go is approximated by a parametric function; critical issues for NDP include the selection of the parametric class of approximating functions, and selection of the approximating parameters.

In this paper, we apply a particular class of NDP algorithms, known as rollout algorithms [2], to risky multi-platform planning and scheduling problems. Rollout algorithms are a form of NDP which exploits knowledge of suboptimal heuristic decision rules to obtain approximations to the optimal cost-to-go for use in NDP. We develop different rollout algorithms for risky multi-platform scheduling, and illustrate the

* This work was supported by the Air Force Office of Scientific Research under contract #F49620-98-C-0023.

relative performance of the rollout algorithms and the original suboptimal decision rules in the context of a specific example. The results illustrate that significant performance improvements can be obtained using rollout algorithms, with a modest increase in computation complexity.

2 Illustrative Overview

To illustrate the types of problems of interest and results developed in this paper, consider the data collection problem illustrated in Figure 1. There are several data collection assets, which may travel to examine targets. There is a value associated with collecting the information on each target. Platforms also run the risk of destruction while performing collection on a asset, due to the presence of local defenses.

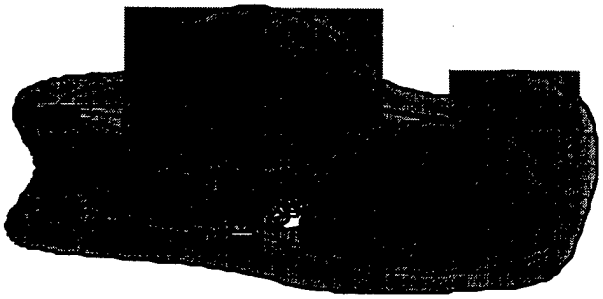


Figure 1 Illustration of Data Collection Problem

Ideally, each data collection asset will be provided a schedule of targets for information collection, which is coordinated among assets to ensure maximal value collected. However, due to the risk inherent in the collection process, platforms can be destroyed, and thus the original schedules should be adapted whenever a destruction event occurs in order to recover the most collection value. If these abrupt events are not anticipated in the original schedules, the possible modifications to the schedules may be so constrained that highly sub-optimal performance results.

The basic theory of dynamic programming provides a framework for developing schedules which anticipate the future occurrence of contingencies such as platform destruction, and hedge the selected schedules in anticipation of needed retasking. Thus, the resulting schedules can be adapted to contingencies with minimal performance degradation, resulting in robust, stable control.

The computational requirements of DP depend on the number of future states required to describe the system. To illustrate the number of states required, assume that there are N targets, M collection assets, and that we simplify physical position descriptions to describe only the N positions of the targets. Then, the number of possible combinations of positions is M^N, and the number of possible uncollected target sets at a given time is 2^N, resulting in numbers of states $(2M)^N$. For modest numbers of assets and targets, the number of states far exceeds our capability for computing and/or storing the resulting optimal decision rules.

Using NDP principles such as rollout strategies greatly reduces the resulting computational complexity. DP considers all of the possible states and computes a tentative decision for each possible state, whereas NDP only computes decisions for states that actually occur in the scenario. Thus, the number of states considered by NDP considered is much smaller, but can only be determined in real-time. In the rollout methodology, once the scenario reaches a given state where a contingency has been observed, new plan options are evaluated in real-time to select the future actions. The result is a practical algorithm for feedback control in complex multi-platform planning and scheduling applications. The fundamental questions about this approach are how good is the performance achieved, and how much real time computation is required. These questions are explored in greater detail in the subsequent sections.

3 Rollout Algorithms

Consider a discrete-time version of a dynamic decision problem,

$$x_{k+1} = f(x_k, u_k, \omega_k)$$

where x_k is the state, u_k is the control to be selected from a finite set $U(x_k)$, and ω_k is a random disturbance. Denote the single-stage cost of control from state and disturbance by $g(x, u, \omega)$.

A control policy $\pi = \{\mu_0, \mu_1, \cdots$ maps, for each stage k, a state x to a control value $\mu_k(x) \in U(x)$. In the N-stage horizon problems considered herein, k takes values , , , and there is also terminal cost $G(x_N)$ that depends on the terminal state x_N. The cost-to-go of policy starting from a state x_j at time k can be computed using the following DP recursion

$$J^{\pi}_k(x) = E\{g(x, \mu_k(x), \omega) + J^{\pi}_{k+1}(f(x, \mu_k(x), \omega))\} \quad (1)$$

for all k and with the initial condition

$$J^{\pi}_N(x) = G(x)$$

The rollout policy based on is denoted by $\tilde{\pi} = \{\tilde{\mu}_0, \tilde{\mu}_1, \cdots\}$, and is defined by the operation

$$\tilde{\mu}_k(x) = \arg\min_{u \in U(x)} E\{g(x, u, \omega) + J^{\pi}_{k+1}(f(x, u, \omega))\} \quad (2)$$

for all x and k. Thus the rollout policy selects decisions by balancing the current cost with future costs-to-go,

where the optimal costs-to-go are approximated by the performance of the base policy τ .

A straightforward approach for computing the rollout control at a given state x and time k is to use Monte Carlo simulations of the base policy. To implement this approach, we consider all possible controls $u \in U(x)$ and generate a "large" number of simulation trajectories of the system starting from x, using u as the first control, and using the policy thereafter. Thus the simulated trajectory has the form

$$x_{i+1} = f(x_i, \mu_i(x_i), \omega_i) \quad i = k+1, \cdots N-1$$

where the first generated state is

$$x_{k+1} = f(x, u, \omega_k)$$

The costs corresponding to these trajectories are averaged to obtain the Q-factor

$$Q(x,u) = E\{g(x,u,\omega) + J^{\pi}{}_{k+1}(f(x,u,\omega))\}$$

In reality, only an approximation $\tilde{Q}(x,u)$ is obtained because of the associated simulation error. The approximation becomes increasingly accurate as the number of simulation trajectories increases. Once the approximate Q-factor $\tilde{Q}(x,u)$ corresponding to each control $u \in U(x)$ is computed, we obtain the approximate rollout control $\tilde{\mu}_k(x)$ by the minimization

$$\tilde{\mu}_k(x) = \arg\min_{u \in U(x)} \tilde{Q}_k(x,u)$$

4 Example: Data Collection Problem

The graph in Figure 2 corresponds to an example data collection problem. Each node represents a geographical area of interest with a one-time value (i.e., data may only be collected once from each location). The arcs represent connectivity among the geographical regions and may be successfully traversed with a known probability. Platforms traverse the graph and collect data (value) at each node, or else they are destroyed while traversing specific arcs. If a platform is destroyed on an arc, the value of the destination node is not collected, which can result in retasking other platforms.

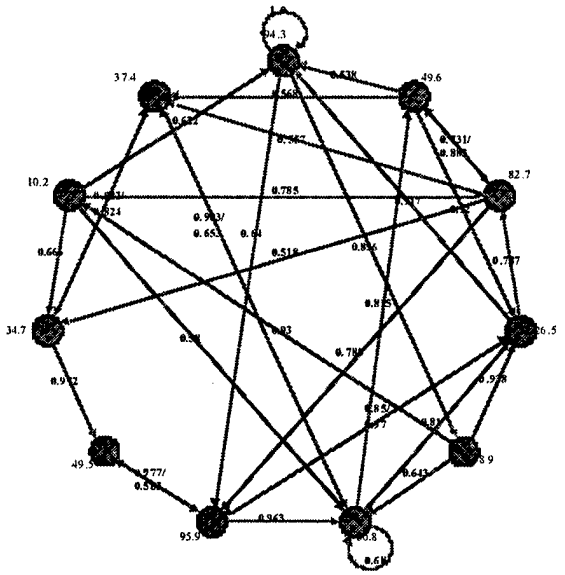


Figure 2 Graph Representation of the Data Collection Problem

The objective is to control the platforms in order to maximize the expected total value collected after N stages ($N = 10$ will be used). Each platform begins at a base node (in this case, node 0 for all platforms) and may traverse one arc during each stage. If a platform does not return to its base node within N stages, there is a penalty associated corresponding to platform loss.

4.1 The Base Policy: Greedy

As a base policy for rollout, we use the greedy policy $\pi = \{\mu_0, \mu_1, \cdots,$ which is defined by the operation

$$\mu_k(x) = \arg\max_{u \in U(x)} E\{g(x,u,\omega)\}$$

for all x and k. The control u is a vector of locations corresponding to the next destination of each platform. Similarly, each element of $\mu_k(x) = [\mu_k^0(x), \mu_k^1(x), \cdots]$ corresponds to a specific platform.

To reduce the computational overhead, we consider the platforms sequentially. The control for the first platform, $\mu_k^0(x)$, is selected independent of the other platforms' controls as:

$$\mu_k^0(x) = \max_{u \in U^0(x)} E\{g(x,u,\omega)\}$$

where $J^0(x)$ are feasible controls for platform 0. The control, $\mu_k^j(x)$, for subsequent platforms is conditioned on all the previously selected controls $\mu_k^0(x), \mu_k^1(x), \cdots, \mu_k^{j-1}(x)$ and defined by the operation

$$\mu_k^j(x) = \max_{u \in U^j(x)} E\{g(x,u,\omega) | \mu_k^0(x), \cdots, \mu_k^{j-1}(x)\}$$

This allows the greedy policy to anticipate the arrival of platforms at specific nodes based on previously selected controls.

The greedy policy also forces platforms to return within N stages by constraining the set $U_k(x)$ of feasible controls to those for which a return within N stages is possible

The performance of the greedy policy corresponds to the cost-to-go from the initial state x_0.

$$J^{\pi}{}_0(x_0) = E\ G(x_N) + \sum_{i=0}^{N-1} g(x_i, \mu_i(x_i), \omega_i)$$

where the expectation is taken over simulation trajectories of the form

$$x_{i+1} = f(x_i, \mu_i(x_i), \omega_i) \quad i = 0, \cdots, N-1$$

The performance of the greedy policy provides a baseline for evaluating the rollout policy.

4.2 Rollout Algorithm

The rollout policy is computed using the greedy policy as its base policy, as indicated in equations (1-2). The performance of the rollout policy is evaluated in a manner similar to the greedy policy, by using the cost-to-go from the initial state x_0.

$$J^{\tilde{\pi}}{}_0(x_0) = E\ G(x_N) + \sum_{i=0}^{N-1} g(x_i, \tilde{\mu}_i(x_i), \omega_i)$$

with the simulation trajectories

$$x_{i+1} = f(x_i, \tilde{\mu}_i(x_i), \omega_i) \quad i = 0, \cdots N-1$$

To reduce the relative variance of performance values, we use the same simulation trajectories in the evaluations of all policies.

One drawback of this approach is that many on-line Monte Carlo simulations may be required to compute the rollout decision at a state. As an alternative, we can use approximations trained with off-line simulations, as discussed in the next subsection.

4.3 Rollouts and Neural Approximations

To reduce the on-line computational overhead of the rollout policies, we propose to train off-line a parametric approximation of the greedy policy performance based on features which characterize the current state. In particular, the features that we use correspond to the values achieved by the greedy policy under a small number of certainty-equivalence scenarios, which capture the graphical dependence of the scheduling problem. This approach was initially proposed in [2].

To compute a feature at a given state x at time k, we fix the remaining disturbances at some nominal values $\bar{\omega}_k, \bar{\omega}_{k+1}, \cdots, \bar{\omega}_{N-1}$, and generate a state and control trajectory of the system using the base policy starting from x_k and time k. The corresponding cost is denoted by $\tilde{J}_k^{\pi}(x_k)$, and is a feature which is used to estimate the true cost $J_k^{\pi}(x_k)$. We use a small number of disturbance trajectories corresponding to different scenarios. The feature values computed for each of these scenarios are combined parametrically to approximate the cost of the base policy using the functional form:

$$\tilde{J}_k(x_k, r) = r_0 + \sum_{m=1}^{M} r_m C_m(x_k) \qquad (3)$$

where $r = (r_0, r_1, \cdots, r_M)$ is a vector of parameters to be determined, and $C_m(x_k)$ is the cost corresponding to the m^{th} scenario. The parameters r are determined by an off-line training process using simulations of the base policy. Equation (3) can then be used on-line, computing the costs $C_m(x_k)$, to evaluate the base policy cost from state x_k at time k.

5 Experimental Results

A series of experiments were performed on the example problem presented in section 4.1, evaluating the performance of the base greedy policy and different variations of rollout algorithms.

The greedy heuristic used for the baseline policy is based on an objective function with two terms, one associated with the achievable value of data collected and the other associated with the potential loss of the vehicle. These values depend on probability ratios associated with risk. The objective function for vehicle k at state x_i is given by

$$g_k(u_i, x_i, \omega) = \frac{p_{ij}}{(1-p_{ij})} n_j(x_i) - \frac{(1-p_{ij})}{p_{ij}} v$$

where $n_j(x)$ is the achievable value of option j given the current state, v is the value of vehicle k, and p_{ij} is the transition probability associated with option j (p_{ij} characterizes the disturbance ω). This objective function is a risk neutral strategy that computes the marginal difference between the largest acceptable loss and the smallest acceptable gain associated with option j.

The greedy heuristic is evaluated by determining the cost-to-go from the initial state x_0.

$$J^{\pi}{}_0(x_0) = E\ G(x_{10}) + \sum_{i=0}^{9} g(x_i, \mu_i(x_i), \omega_i)$$

where the expectation is approximated with 100 Monte Carlo simulation trajectories of the form

$$x_{i+1} = f(x_i, \mu_i(x_i), \omega_i) \quad i = 0, \cdots, 9$$

The evaluation of the greedy heuristic resulted in an estimated value of 569.3 (the standard deviation of the estimate is 12.2). As a benchmark, the total value achievable is 714, arising from:

Total Collectible Value	610
Total Vehicle Value	104

We conducted three types of experiments, exploring different rollout options. The first set of experiments used different number of Monte Carlo runs in the rollout algorithm to evaluate the relative performance of the different controls for each state considered. Tested conditions ranged from 5 to 40 Monte Carlo experiments per decision.

The second set of experiments evaluated alternatives in the planning horizon considered in the rollout problem. It has been conjectured [2] that the performance of rollout strategies degrades after increasing the planning horizon beyond a threshold, due to the approximation of the optimal future policy by a base policy. This approximation becomes less accurate with increasing planning horizon. To test this, we conducted experiments where we varied the horizon used by the rollout policy to evaluate the base policy.

The final set of experiments compares the performance of the Monte Carlo rollout algorithms with the performance of the algorithms based on parametric function approximations using certainty equivalence features.

5.1 Variations in Monte Carlo Runs

In these experiments, the number of Monte Carlo runs used to evaluate the performance of the base policy in the rollout algorithm varies from 5, 10, 20, 30, and 40 Monte Carlo runs. For each of these experiments, evaluation of the rollout policy performance is conducted in a manner identically to the evaluation of the greedy policy performance described previously. That is, 100 independent Monte Carlo simulation trajectories are used, of the form

$$x_{i+1} = f(x_i, \pi_i(x_i), \omega_i) \quad i = 0, \cdots 9$$

where $\pi_i(x_i)$ is the rollout control policy defined by the operation

$$\pi_i(x) = \arg\max_{u \in U(x)} E\{g(x,u,\omega) + J^{\pi}{}_{i+1}(f(x,u,\omega))\}$$

and this last expectation is also approximated with either 5, 10, 20, 30, and 40 Monte Carlo simulations.

The results of this experiment are shown in Figure 3. As the results indicate, the performance achieved by the rollout strategies using 20 or more Monte Carlo simulations range on average from 600 to 610, a range which is far superior to the greedy policy performance average of 569.3. The results suggest that a modest number of simulations are required to select good controls in this example.

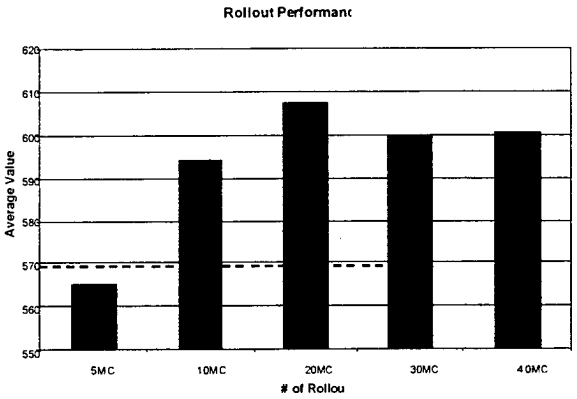


Figure 3 Rollout Performance

5.2 Variations in Planning Horizon

The rollout algorithm performs a single policy improvement step on the greedy heuristic. This allows the rollout trajectory to deviate significantly from the greedy trajectory, especially over long horizons. In this case, the cost-to-go estimate derived from greedy trajectories may not reflect the actual cost-to-go of the rollout algorithm. One way of avoiding this problem is to evaluate the future cost-to-go of the base policy over a limited horizon. Figure 4 shows the performance of this rollout algorithm with various horizons, where 20 Monte Carlo experiments are used to evaluate each policy.

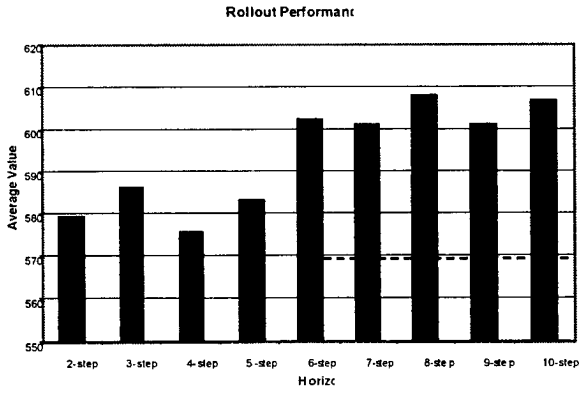


Figure 4 Performance as a Function of Horizon

The results in Figure 4 do not support the conclusion that there is a maximum planning horizon beyond which the rollout performance degrades. However, the results need closer examination to understand whether the use of a different base heuristic would exhibit similar behavior.

5.3 Off-Line Training vs Monte Carlo

In these experiments, we compare the performance of rollout algorithms based on the parametric approximations of Section 4.3 with the Monte Carlo rollout algorithms of Section 4.2. The parametric approximations were based on certainty equivalent features, which corresponded to selecting specific threshold values and declaring all arcs with probabilities of survival greater than the threshold to be safe, and all arcs with probabilities of survival less than or equal to the threshold to have a certainty of destroying any vehicles on those arcs. The resulting graph is a deterministic graph, which leads to fast evaluation of the base policy. The performance obtained for different values of thresholds provided the base features for the parametric approximation.

Several rollout algorithms were evaluated: First, we used algorithms based on single features, with trivial parametric approximation. Second, we used algorithms using weighted combinations of features, with weights trained off-line using training data. Finally, we used optimized weights, searching in the space of possible weights for optimal performance; this is not a practical algorithm, but provides a baseline for the achievable performance from training algorithms.

The experimental results are summarized in Figure 5. These experiments used only two features in the parametric interpolation, corresponding to two different values of thresholds. In Figure 5, four pairs of features are considered along the x-axis. For each pair of features, the rollout performance is evaluated with optimal weights, and with trained weight. Rollout performance is also evaluated for each of the features used in isolation.

In Figure 5, the upper dashed line represents the best performance achieved using the Monte Carlo rollout approach, and the lower dashed line represents the performance of the greedy heuristic. The dotted line indicates the performance using two statistically determined features combined with equal weights.

The results of Figure 5 are surprising, in that the algorithms based on off-line training seldom approach the performance achieved by the optimal weighted combination. Figure 5 shows that the rollout algorithm based on features with trained weights was not able to offer a consistent and significant improvement over the greedy heuristic. In some cases the combination of features with trained weights were not able to perform as well as the features individually.

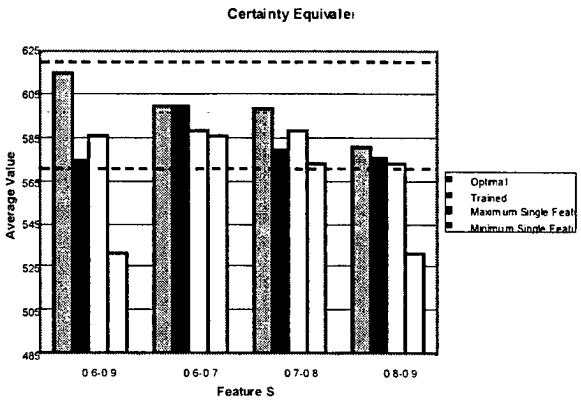


Figure 5 Rollout Performance with Feature Pairs

Figure 5 shows that the optimally selected weights using the features 0.6 and 0.9 achieved an overall performance close to that of the Monte Carlo approach. However, when other pairs of features were used, the performance was significantly worse. An alternative to training or optimization is to select the parameters analytically. The dotted line in Figure 5 shows the performance of a combination of features that were selected with equal weights to "match" the statistical distribution (mean and variance) of risk within the problem. This approach provides a significant improvement over the greedy heuristic without the computational cost of training or optimization. This approach appears worthy of further investigation due to its simplicity.

In sum, rollout algorithms using parametric approximations did not perform as well as rollout algorithms using Monte Carlo simulations.

6 Conclusions

In this paper we have considered the use of rollout algorithms for adaptive multi-platform scheduling in a risky environment. We explored different variations of rollout algorithms, using combinations of on-line Monte Carlo simulation and parametric approximations. Our experimental results show that rollout algorithms using on-line simulation perform significantly better than the reference base heuristic policies, using only a modest number of Monte Carlo trajectories.

In our experiments, we found that rollout algorithms based on parametric approximations to the cost-to-go failed to achieve the level of performance of similar rollout algorithms using on-line Monte Carlo simulations. The parametric approximations suffered from two limitations: First, the training techniques often failed to identify the best weight combinations. Second, the parametric approximations were unable to generalize accurately across the broad class of states

which occurred in the problem. Our experiments were limited to simple classes of parametric approximations using the concept of certainty equivalence scenarios. Exploration of alternative approximations using different features is an area for future investigations.

The main limitation of the Monte Carlo rollout algorithms is the amount of on-line computation required to evaluate the different options at each state. We are currently investigating techniques based on discrete-event systems and perturbation analysis [4] to reduce the number of simulations required to evaluate multiple alternatives.

7 References

[1] Bertsekas. D.P.. Tsitsikis. J.N.. *Neuro-Dynamic Programming*. Athena Scientific. 1996.

[2] Bertsekas. D.P.. Castañon. D.A.. "Rollout Algorithms for Stochastic Scheduling Problems." *Journal of Heuristics*. V. 5. 1999.

[3] Bertsekas. D.P.. *Dynamic Programming and Optimal Control*. Athena Scientific. 1995

[4] Ho. Y.C.. Cassandras. C.G.. Chen. C.H.. Dai. L.. "Ordinal Optimization and Simulation." submitted to special issue on simulation to be published by INFORMS 1999.

[5] Ross. S. M.. *Introduction to Stochastic Dynamic Programming*. Academic Press. N.Y.. 1983.

Enterprise Engineering: A Framework for Optimal Control of Hybrid Dynamical Systems

Y. Wardi
Georgia Institute of Technology
School of Electrical & Computer Engineering
Atlanta, GA 30332
ywardi@ece.gatech.edu

C.G. Cassandras
Boston University
Department of Manufacturing Engineering
Boston, MA 02215
cgc@enga.bu.edu

Abstract

This paper presents a model-based approach to cross-layer control of enterprise systems. In particular, it focuses on vertical integration between the physical layer and the discrete-event layer in hybrid dynamical systems. A general modeling paradigm for optimal control is discussed. Its analysis, based on the special structure of hybrid dynamical systems, leads to the development of low-complexity algorithms for computing optimal controls.

Key words. Enterprise control, hybrid systems, optimal control.

1 Introduction

Enterprise systems are hierarchical dynamical systems with a vertical command structure across layers and distributed subsystems within layers. Such systems typically have three layers: physical layer, discrete-event layer and strategic layer. At the physical layer there are dynamical systems whose states represent mechanical, chemical, or other physical characteristics. These dynamics are time driven and are described by differential equations or difference equations. The discrete-event layer is concerned with synchronization and co-ordination of various tasks and activities performed by the physical-layer subsystems. Its dynamics are event driven, and its state includes temporal parameters such as tasks' starting times and durations. The dynamic evolution of this state often is described by timed Discrete Event Dynamic Systems (DEDS) like Petri nets, state automata, max-plus algebras, or queueing networks. The strategic layer concerns global issues and long-term strategic decisions.

The enterprise-system paradigm includes common systems in various technological areas, like manufacturing and C^4I, and although this talk will focus on manufacturing applications, the greater generality of its scope will become evident. In a manufacturing enterprise the physical layer concerns physical and material changes associated with production. The discrete-event layer considers logistical issues associated with the movement and storage of parts and products, while the strategic layer is concerned with global financial, political, societal and other long-term issues.

Enterprise systems can be of a very large scale, and their dynamics, associated with the above three layers, are of different time scales. It is a major challenge to characterize the interaction among the three layers in ways that enable the development of unified, comprehensive tools for design, planning and control. Most of the existing tools consider the dynamics at each layer separately, without explicitly including inter-layer information flow in the dynamic model. What is needed is a general paradigm that clearly defines the interfaces between the layers, and thus can serve as a basis for developing integrated control architectures.

This paper addresses the question of integration between the physical layer and the discrete-event layer and it focuses on the development of a model-based framework for optimal control. The underlying system has the structure of a Hybrid Dynamical System (HDS), namely a bi-layer hierarchical system with time-driven dynamics at the lower layer and event-driven dynamics at the upper layer. Such systems can be viewed as extensions of Discrete Event Dynamic Systems (DEDS) to include time-driven dynamics [6, 2, 10, 5, 8, 11]. The dynamics at the discrete-event layer typically are described by a timed-DEDS model, while the dynamics at the physical layer are described by differential equations or difference equations. The schedule of states' transitions, or events' occurrences at the upper layer is determined by the state variables associated with the timed-driven dynamics at the lower layer. It is this connection between the two layers that defines the hybrid nature of the HDS.

Suppose that there is a control parameter associated with the physical state, and hence having an effect on the schedule of state transitions at the discrete-event layer. Such control parameters have an indirect im-

pact on the event-driven dynamics. Various applications in manufacturing and C^4I involve performance metrics that are functions of the state associated with the discrete-event dynamics, and especially the timing of events' occurrences. The optimal control problem consists of minimizing such a performance metric by an appropriate choice of the control parameters. Note the hybrid aspect of the problem: the performance measure is expressed in terms of the state of the DEDS while the control variables are parameters of the physical layer.

Following a general formulation of the optimal control problem, the paper considers numerical methods. To date, only single-stage systems have been investigated, and the derived results will be presented below. However, these results reveal a considerable special structure that might be extendable to more general networks and systems. The results are presented here without proofs, which can be found in [11].

Consider a single-stage manufacturing system modeled as a single-server queue whose objective is to process N given jobs, which are denoted by C_i, $i = 1, \ldots, N$. The jobs' service order is assumed to be fixed and indicated by the index i: C_{i+1} is processed after C_i. The queue is assumed to have an infinite buffer capacity. The control variables constitute parameters of the service times, and the performance metric is a function of the completion (departure) times and the control parameters.

The dynamics of the jobs' processing are time driven, and have the following form (see [4, 5]). Let $z_i \in \mathbb{R}^n$ denote the physical state associated with the processing of C_i, and suppose that it evolves according to the following differential equation,

$$\dot{z}_i = g_i(z_i, u_i, t), \quad z_i(\tau_i) = \zeta_i \tag{1.1}$$

where τ_i is the time processing begins and ζ_i is the initial state at that time. The control variable u_i is used to attain a final desired physical state corresponding to a target "quality level." Specifically, if the service time for the ith job is $s_i(u_i)$ and $\Gamma_i(u_i) \subset \mathbb{R}^n$ is a given set, then the control u_i is chosen to satisfy the stopping rule

$$\begin{aligned} s_i(u_i) &= \min\{t \geq 0 : z_i(\tau_i + t) \\ &= \int_{\tau_i}^{\tau_i + t} g_i(z_i, u_i, \nu)d\nu + \zeta_i \in \Gamma_i(u_i)\} \end{aligned} \tag{1.2}$$

where u_i takes on a fixed constant value during the interval $[\tau_i, \tau_i + t)$, and the "min" is assumed to exist. On the other hand, the *temporal* state of the ith job is denoted by x_i and represents the completion time of C_i at the server. With a_i denoting the arrival time of C_i, the event-driven dynamics describing the evolution of the temporal state are given by the following "max-plus" recursive equation:

$$x_i = \tau_i + s_i(u_i) = \max\{a_i, x_{i-1}\} + s_i(u_i), \tag{1.3}$$

where we set $x_0 = -\infty$ to ensure that $\tau_1 = a_1$, namely that the first job begins service as soon as it arrives. It is assumed that the job arrival sequence $\{a_1, \ldots, a_N\}$ is given.

Note that the two layers of the above hybrid system are coupled through the choice of the control parameters u_i, $i = 1, \ldots, N$. Given a cost function $L_i(x_i, u_i)$ associated with the processing of C_i and its completion time, the optimal control problem considered here has the form,

$$\min_{u_1, \ldots, u_N} J = \sum_{i=1}^{N} L_i(x_i, u_i) \tag{1.4}$$

subject to Eqns. (1.1), (1.2) and (1.3). Note that this formulation does not require an explicit cost on the physical state z_i, since (1.2) ensures that each job satisfies a given quality requirement, i.e., $z_i(\tau_i + s_i(u_i)) = z_i(x_i) \in \Gamma_i(u_i)$.

As an example of the ith-stage cost function $L_i(x_i, u_i)$, suppose that u_i represents a quality parameter associated with production, and let $\theta_i(u_i)$ be a monotone-nonincreasing function reflecting on a higher cost for lower-quality products. On the other hand, let $\psi_i(x_i)$ be a convex, positive-valued function having its minimum at a point $\delta_i \in \mathbb{R}$, where δ_i is a given due date for service completion of C_i. Let $L_i(x_i, u_i) = \psi_i(x_i) + \theta_i(u_i)$. Observe that the first part of the cost L_i is a cost measure of the discrepancy between the completion time of C_i and its target due date, while the second part measures the product's quality. The control variable u_i affects $\psi_i(x_i)$ indirectly through its impact on the completion time x_i. This example highlights a situation that often arises in real-world manufacturing applications: process engineers strive to manufacture high-quality products, while management is mainly concerned with the product's delivery schedule. Conflicts naturally arise, and their resolution at either the planning or operation phases requires an integrated approach to planning and control.

The optimal control program in Eq. (1.4) generally is nondifferentiable due to the "max" term in (1.3), nor can it be expected to be convex. Yet it possesses considerable special structure that makes it possible to apply established techniques from the theory of optimal control of timed-driven systems [1, 3] to the present case of HDS. In particular, developments of the variational principle [9] lead to a strong necessary and sufficient optimality condition [5], and to the development of low-complexity algorithms for computing the optimal controls [11]. The first-order necessary optimality condition, based on an application of the calculus of variations, will be shown to be satisfied by a unique control sequence. This result is somewhat surprising in view of the absence of convexity. The optimization algorithm that will be presented is based on a decom-

position of the program in (1.4) into a sequence of convex, differentiable subproblems. Although the number of these subproblems appears to be of the order of 2^N, the special structure of the dynamic system results in a reduction to the order of N.

Section 2 formulates the optimal control problem and presents an optimality condition that is both necessary and sufficient. Section 3 discusses the algorithm and its convergence properties, and Section 4 concludes the paper.

2 Problem Formulation and Optimality Conditions

Consider first a general optimal control problem defined on a discrete-time dynamical system of the following form,

$$x_i = f_i(x_{i-1}, u_i), \quad i = 1, \ldots, N; \tag{2.1}$$

here $u_i \in \mathbb{R}$ is the *ith*-stage control and $x_i \in \mathbb{R}$ is the *ith*-stage state, and the initial state x_0 and the final time N are given. The control and state variables are one-dimensional for the sake of exposition's simplicity, and extensions to vectors are straightforward. Let $L_i(x_i, u_i)$ be an *ith*-stage, real-valued cost measure, and consider the following performance functional, J,

$$J := J(\underline{u}) = \sum_{i=1}^{N} L_i(x_i, u_i), \tag{2.2}$$

where $\underline{u} := (u_1, \ldots, u_N) \in \mathbb{R}^N$ constitutes the control sequence. The optimal control problem, denoted by P, has the following form,

$$P: \quad \min\{J(\underline{u}) : \underline{u} \in \mathbb{R}^N\}. \tag{2.3}$$

The standard first-order optimality condition is the stationarity condition

$$\frac{dJ}{du_i}(\underline{u}) = 0, \quad i = 1, \ldots, N, \tag{2.4}$$

provided that these derivatives exist and are continuous functions of $\underline{u}$. The partial derivatives $\frac{dJ}{du_i}$ can be expressed in terms of the costates λ_i, $i = 1, \ldots, N$, computed backwards recursively; see [1, 3] for this fundamental result of optimal control theory:

$$\lambda_i = \frac{\partial f_{i+1}}{\partial x}(x_i, u_{i+1})\lambda_{i+1} + \frac{\partial L_i}{\partial x}(x_i, u_i), \quad i = N-1, \ldots, 1, \tag{2.5}$$

with the boundary condition $\lambda_N = \frac{\partial L_N}{\partial x}(x_N, u_N)$; and,

$$\frac{dJ}{du_i} = \frac{\partial f_i}{\partial u}(x_{i-1}, u_i)\lambda_i + \frac{\partial L_i}{\partial u}(x_i, u_i). \tag{2.6}$$

Thus, solving the optimal control problem P (2.3) to the extent of computing a control-sequence $\underline{u}$ satisfying the first-order optimality condition, amounts to solving the two-point boundary-value problem (TPBVP) given by Eqs. (2.1), (2.5) and the following Eq. (2.7),

$$\frac{\partial f_i}{\partial u}(x_{i-1}, u_i)\lambda_i + \frac{\partial L_i}{\partial u}(x_i, u_i) = 0, \quad i = 1, \ldots, N. \tag{2.7}$$

Consider now the special case of single-stage manufacturing systems, as earlier mentioned. The state equation (2.1) assumes the form of (1.3), namely,

$$x_i = f_{i-1}(x_{i-1}, u_i) = \max\{a_i, x_{i-1}\} + s_i(u_i). \tag{2.8}$$

In order to simplify the exposition, we further identify the control parameter with the processing time, namely, $s_i = u_i$. To justify this, recall that $u_i \in \mathbb{R}$ is a parameter of the service time s_i of job C_i. Often, the function $s_i(\cdot)$ is either monotone increasing or monotone decreasing, and hence it has an inverse. The forthcoming derivations lead to the computation of the optimal controls through the service times s_i, and hence the corresponding control variables can be computed directly from the relation $s_i = s_i(u_i)$. Therefore, to simplify the exposition, we identify the control variables with the service times, i.e., we set $s_i = u_i$, and carry out the rest of the discussion in terms of the notation u_i.

Concerning the *ith*-stage cost functions, let us suppose that $L_i(x_i, u_i)$ is separable, i.e.,

$$L_i(x_i, u_i) = \theta_i(u_i) + \psi_i(x_i), \tag{2.9}$$

where $\theta_i : \mathbb{R}^+ \to \mathbb{R}^+$ and $\psi_i : \mathbb{R} \to \mathbb{R}^+$ are given continuously differentiable functions. The function $\theta_i(u_i)$ acts as a "quality" measure associated with the processing of C_i. Frequently longer processing times correspond to higher quality, while cost functions tend to penalize lower quality more heavily. Therefore, we have the function θ_i be monotone decreasing. Regarding the state-dependent cost function, $\psi_i(x_i)$ is assumed to be convex. Its minimum point, δ_i, can be representative of a target due date, and $\psi_i(x_i)$ can reflect on a cost related to the discrepancy between the actual completion time (x_i) and the target due date (δ_i). For example, $\psi(x_i) = (x_i - \delta_i)^2$. The following assumptions concerning the cost functions will be made throughout, they greatly simplify the analysis without much loss of generality.

Assumption 2.1. For every $i = 1, \ldots, N$, the function $\theta_i : \mathbb{R}^+ \to \mathbb{R}^+$ is twice continuously differentiable, strictly convex, the derivative $\theta_i'(s)$ is negative-valued for every $s > 0$, and the following limits hold,
$\lim_{s \to 0^+} \theta_i(s) = -\lim_{s \to 0^+} \theta_i'(s) = \infty$,
$\lim_{s \to \infty} \theta_i(s) = \lim_{s \to \infty} \theta_i'(\theta) = 0$.
As an example, $\theta_i(s) = s^{-1}$.

Assumption 2.2. For every $i = 1,\ldots,N$, the function $\psi_i : \mathbb{R} \to \mathbb{R}^+$ is twice continuously differentiable and strictly convex, and its minimum is obtained at a finite point δ_i.

For example, $\psi_i(x) = (x - \delta_i)^2$ for a given $\delta_i \in \mathbb{R}$.

Let us next consider the costate equation (2.5) and the stationarity condition given by Eq. (2.7). By Eq. (2.8), the partial derivatives of $f_i(x_{i-1}, u_i)$ are,

$$\frac{\partial f_i}{\partial u}(x_{i-1}, u_i) = 1, \tag{2.10}$$

and

$$\frac{\partial f_i}{\partial x}(x_{i-1}, u_i) = \begin{cases} 1, & \text{if } x_{i-1} > a_i \\ 0, & \text{if } x_{i-1} < a_i. \end{cases} \tag{2.11}$$

This partial derivative does not exists whenever $x_{i-1} = a_i$, namely, when C_{i-1} departs at the time C_i arrives to the queue. Regarding the partial derivatives of the cost function, observe, by Eq. (2.9), that

$$\frac{\partial L_i}{\partial u}(x_i, u_i) = \theta_i'(u_i), \tag{2.12}$$

while

$$\frac{\partial L_i}{\partial x}(x_i, u_i) = \psi_i'(x_i). \tag{2.13}$$

Note the usage of "prime" to denote derivative with respect to the appropriate variable in the last two equations.

We next define some structural properties associated with the trajectory of a control sequence $\underline{u} = (u_1,\ldots,u_N)$. Recall that $x_0 = -\infty$ by convention, and similarly we define $a_{N+1} = \infty$.

Definition 2.1. A job C_i is *critical* if it departs at the arrival time of the next job, i.e., $x_{i-1} = a_i$.

Definition 2.2. A *block* is a contiguous set of jobs, $C_k,\ldots,C_n$, for $1 \le k \le n \le N$ such that the following three conditions are met.

1. $x_{k-1} < a_k$;
2. $x_n < a_{n+1}$;
3. For every $i = k\ldots,n-1$, $x_i \ge a_{i+1}$.

We make some observations. Criticality corresponds to nondifferentiability. In fact, in the absence of critical jobs, the cost functional $J(\underline{u})$ is continuously differentiable. A block corresponds to a busy period of the server. If the job C_i is critical than it and the next job, C_{i+1}, are in the same block.

Definition 2.3. A *block structure* is a partition of the jobs $C_1,\ldots,C_N$ into blocks.

For every $i = 1,\ldots,N$, we denote by $n(i)$ the index of the last job in the block containing C_i, that is,

$$n(i) = min\{n \ge i : x_n < a_{n+1}\}. \tag{2.14}$$

Moreover, let $m(i)$ denote the index of the next job starting at C_i that either is critical or it ends the block containing C_i. That is,

$$m(i) = \min\{m \ge i : x_m \le a_{m+1}\}. \tag{2.15}$$

Suppose momentarily that there are no critical jobs, and consider the costate equation (2.5) and the following equation (2.6) for the partial derivative $\frac{dJ}{du_i}$. By Eq. (2.5) together with (2.11) and (2.14), we obtain,

$$\lambda_i = \sum_{j=i}^{n(i)} \psi_j'(x_j), \tag{2.16}$$

and by Eqs. (2.6), (2.10) and (2.12),

$$\frac{dJ}{du_i} = \lambda_i + \theta_i'(u_i). \tag{2.17}$$

Let us define the notation $\xi_{i,n}$, for every (i,n) such that $1 \le i \le n \le N$, by

$$\xi_{i,n} = \theta_i'(u_i) + \sum_{j=i}^{n} \psi_j'(x_j). \tag{2.18}$$

By Eqs. (2.17), (2.16) and (2.18) we have that $\frac{dJ}{du_i} = \xi_{i,n(i)}$, and the first-order necessary optimality condition assumes the following form.[1]

Fact 2.1. If there are no critical jobs associated with the optimal control $\underline{u}$, then, for all $i = 1,\ldots,N$,

$$\xi_{i,n(i)} = 0. \tag{2.19}$$

Consider next the possibility of critical jobs. As earlier remarked, the cost functional $J(\underline{u})$ might be nondifferentiable. However, we have the following intuitive result.

Fact 2.2. The one-sided derivatives of J with respect to u_i exist, and are equal to the following respective terms.

$$\frac{dJ}{du_i^-} = \xi_{i,m(i)}, \quad \text{and} \quad \frac{dJ}{du_i^+} = \xi_{i,n(i)}. \tag{2.20}$$

The first-order necessary optimality condition becomes,

$$\text{For all } i = 1,\ldots,N, \quad \xi_{i,m(i)} \le 0 \quad \text{and} \quad \xi_{i,n(i)} \ge 0. \tag{2.21}$$

[1] As earlier mentioned, all proofs are relegated to [11].

Observe that (2.21) constitutes an extension of (2.19) from the differentiable case where $m(i) = n(i)$ to the nondifferentiable case involving the presence of critical jobs.

Fact 2.2 has been derived by direct variational arguments. The following optimality condition is considerably stronger, due to the second part of the next assertion.

Fact 2.3. If the control sequence $\underline{u}$ is optimal, then the following two conditions hold.

1. For all $i = 1, \ldots, N$, $\xi_{i,m(i)} \leq 0$ and $\xi_{i,n(i)} \geq 0$.
2. For every $i = 1, \ldots, N$, and for every $\nu = i, \ldots, m(i)$, $\xi_{i,m(i)} = \xi_{\nu,m(i)}$.

We remark that, in the differentiable case where $m(i) = n(i)$, the latter assertion is immediately satisfied with $\xi_{i,m(i)} = \xi_{\nu,m(i)} = 0$.

The above optimality condition is not only necessary but is sufficient as well, as can be seen in the following result.

Fact 2.4. There exists a unique control sequence $\underline{u} = (u_1, \ldots, u_N)$ such that, the optimality condition in Fact 2.3 is satisfied.

3 Algorithm for Computing Optimal Controls

The functional $J(\underline{u})$ is generally neither convex nor differentiable, and consequently, a general-purpose algorithm, based on nondifferentiable calculus, for computing its minimum may require overwhelming computing resources. There is, however, enough special structure that can be exploited to simplify the computation. Specifically, the solution of the optimal control problem P (2.3) will be computed by solving a finite sequence of differentiable, convex programming problems.

To set the stage, let us embed the problem P within a family of optimal control problems, denoted each by $P_{k,n}$, where the two-dimensional index (k,n) ranges over the set $\{1 \leq k \leq n \leq N\}$. For every such double-index (k,n) the problem $P_{k,n}$ is the same as the problem P in (2.3) except that only the jobs $C_k, \ldots, C_n$ are present. In other words, the dynamics of the system are given by the equation

$$x_i = \max\{x_{i-1}, a_i\} + u_i, \qquad i = k, \ldots, n, \qquad (3.1)$$

with the boundary condition $x_{k-1} = -\infty$, and the cost functional, denoted by $J_{k,n}$, is defined by

$$J_{k,n}(u_k, \ldots, u_n) = \sum_{i=k}^{n} [\theta_i(u_i) + \psi_i(x_i)]. \qquad (3.2)$$

The corresponding optimal control problem is

$$P_{k,n}: \quad \min\{J_{k,n}(u_k, \ldots, u_n) : u_i \geq 0, \quad i = k, \ldots, n\}. \qquad (3.3)$$

Observe that the original problem P is identical to $P_{1,N}$.

Now further consider an index (k,n) as above, namely $1 \leq k \leq n \leq N$, and define the function $\tilde{J}_{k,n}(u_k, \ldots, u_n)$ by

$$\tilde{J}_{k,n}(u_k, \ldots, u_n) = \sum_{i=k}^{n} [\theta_i(u_i) + \psi_i(a_k + \sum_{j=k}^{i} u_j)]. \qquad (3.4)$$

Lemma 3.1. The function $\tilde{J}_{k,n}$ is continuously differentiable and strictly convex.

Consider next the following program, denoted by $Q_{k,n}$:

$$Q_{k,n}: \quad \min\{\tilde{J}_{k,n}(u_k, \ldots, u_n) : u_i \geq 0, \; i = k, \ldots, n; \; a_k + \sum_{j=k}^{i} u_j \geq a_{i+1}, \; i = k, \ldots, n-1\}. \qquad (3.5)$$

Since the cost function $\tilde{J}_{k,n}$ is strictly convex and the constraints are linear, and since by Assumption 2.2 the functions ψ_i attain their minima at finite points, the program $Q_{k,n}$ has a unique solution at a finite point. Moreover, this solution must satisfy the inequalities $u_i > 0$, $i = k, \ldots, n$, in view of Assumption 2.1.

Lemma 3.2. If the jobs $C_k, \ldots, C_n$ constitute a single block according to the solution point (i.e., optimal control sequence) of the program $P_{k,n}$, then the solution points of $P_{k,n}$ and $Q_{k,n}$ are identical.

An implication of Lemma 3.2 is that, if the block structure of the optimal control $\underline{u} = (u_1, \ldots, u_N)$ of the problem P were known in advance, then the computation of the optimal controls could proceed by solving a differentiable, convex program for each block. The problem is, of course, that the block structure of the optimal solution is not known in advance. This suggests that the search for the solution for P (2.3) be based on scanning the various block structures of the jobs $C_1, \ldots, C_N$. Suppose we try a block structure consisting of blocks $B_1, \ldots, B_M$ for some $M \in \{1, \ldots, N\}$, and suppose that the mth block, B_m, consists of the jobs $C_{k(m)}, \ldots, C_{n(m)}$. To count the blocks and the jobs in a contiguous fashion we say that the block structure is $\{B_1, \ldots, B_M\}$;

for every $m = 1,\ldots,M$ $B_m = \{C_{k(m)},\ldots,C_{n(m)}\}$; $k(1) = 1$; $k(m) = n(m-1)+1$ for all $m = 2,\ldots,M$; and $n(M) = N$. Consider such a block structure, and solve the convex programs $Q_{k(m),n(m)}$, $m = 1,\ldots,M$. If the solutions preserve the block structure, namely $x_{n(m)} < a_{k(m+1)}$ for all $m = 1,\ldots,M-1$, then the control sequence consisting of concatenation of the solutions of the above convex programs constitutes a solution for P as well. If the block structure is not preserved by the above solutions then the search has to continue with another block structure. The main difficulty with this approach is in its exponential complexity: there are 2^{N-1} different block structures.

To ameliorate this problem we exploit the special structure to reduce the search space to the size of $2N-1$. The idea is based on the following assertion.

Theorem 3.1. For every $k = 2,\ldots,N$, every block associated with the solution of $P_{k,N}$ constitutes a subset of a block associated with the solution of $P_{k-1,N}$.

This theorem has some implications to algorithm development. Suppose we have solved the program $P_{k,N}$ for some $k = 2,\ldots,N$, and the associated block structure is $\{\bar{B}_1,\ldots,\bar{B}_M\}$. Next, we add the job C_{k-1}, and consider the program $P_{k-1,N}$. The blocks resulting from its solution must constitute concatenations of entire blocks from the set $\{\bar{B}_1,\ldots,\bar{B}_M\}$. In other words, blocks can be merged but cannot be broken down in passing from $P_{k,N}$ to $P_{k-1,N}$. Let $\{B_1,B_2,\ldots,\}$ constitute the block structure associated with the solution of $P_{k-1,N}$. Then B_1 must be comprised of either one of the following: $B_1 = \{C_{k-1}\}$; $B_1 = \{C_{k-1}\}\cup\bar{B}_1$; $B_1 = \{C_{k-1}\}\cup\bar{B}_1\cup\bar{B}_2$; etc.. Let us define $\bar{B}_0 = \{C_{k-1}\}$. Then, for some $m \in \{1,\ldots,M\}$,

$$B_1 = \cup_{j=0}^{m}\bar{B}_j. \tag{3.6}$$

If $m = M$ then the block structure associated with $P_{k-1,N}$ constitutes a single block. Moreover, if $m < M$ then $B_2 = \bar{B}_{m+1}$, $B_3 = \bar{B}_{m+2}$ (if $m < M-1$), etc.. In fact, none of the jobs following the block $\bar{B}_m$ has been affected by the inclusion of the additional job C_{k-1}. The remaining question is how to compute m.

Let us denote the optimal control sequence of the problem $P_{k,N}$ by $\bar{u}_k,\ldots,\bar{u}_N$, and the resulting state trajectory by $\bar{x}_k,\ldots,\bar{x}_N$. Recall that the corresponding block structure is $\{\bar{B}_1,\ldots,\bar{B}_M\}$, where for every $j = 1,\ldots,M$, $\bar{B}_j = \{C_{k(j)},\ldots,C_{n(j)}\}$. Likewise, let us denote by $u_{k-1},\ldots,u_N$ the solution sequence (optimal controls) of $P_{k-1,N}$, and the associated state trajectory by $x_{k-1},\ldots,x_N$, and recall that $B_1 = \cup_{j=0}^{m}\bar{B}_j$ for some $m \in \{1,\ldots,M\}$, and with $\bar{B}_0 := \{C_{k-1}\}$. For all $i > n(m)$, $u_i = \bar{u}_i$. Our objective is to compute m. Consider first the possibility that $m = 0$, i.e., $B_1 = \{C_{k-1}\}$. Solve $P_{k-1,k-1}$ to yield u_{k-1}, and compute the corresponding state variable x_{k-1} via Eq. (3.1). If $x_{k-1} < a_k$ then we are done, with $B_1 = \bar{B}_0$. If, on the other hand, $x_{k-1} \geq a_k$, then we know that $m \geq 1$. We next try $m = 1$, namely $B_1 = \bar{B}_0 \cup \bar{B}_1$. Solving $P_{k-1,n(1)}$ we compute the resulting controls $u_{k-1},\ldots,u_{n(1)}$, and the corresponding state variable $x_{n(1)}$ (via (3.1)). If $x_{n(1)} < a_{n(1)+1}$ then we are done, with $m = 1$. If, on the other hand, $x_{n(1)} \geq a_{n(1)+1}$ then we try $m = 2$ in a similar fashion, etc..

To formalize, the following algorithm computes the solution to $P_{k-1,N}$ from a solution to $P_{k,N}$.

Algorithm 3.1.
Begin {
Initialize: Set $m = 0$ and set $B_1 = \bar{B}_0 = \{C_{k-1}\}$.
Solve $Q_{k-1,k-1}$ to compute u_{k-1}, and compute $x_{k-1} = a_{k-1} + u_{k-1}$.
Set $k(0) = n(0) = k-1$.
Main loop:
While $(x_{n(m)} \geq a_{n(m)+1})$ {
Set $B_1 = B_1 \cup \bar{B}_{m+1}$.
Solve the program $Q_{k-1,n(m+1)}$ to compute $u_{k-1},\ldots,u_{n(m+1)}$,
and compute $x_{n(m+1)} = a_{k-1} + \sum_{i=k-1}^{n(m+1)} u_i$.
Set $m = m+1$.
}
Set $u_i = \bar{u}_i$ for all $i = n(m)+1,\ldots,N$.
} end.

We remark that the "while" loop must be entered at most M times. To see this observe that, if $m = M$ then $n(m) = N$, and since $a_{N+1} = \infty$ by definition, the condition of the "while" loop is not satisfied.

The following procedure solves the optimal control problem P.

Algorithm 3.2.
Begin {
Initialize: Solve $P_{N,N}$ by solving $Q_{N,N}$. Set $k = N$.
Main loop:
While $(k > 1)$ {
Solve $P_{k-1,N}$ by using Algorithm 3.1.
Set $k = k-1$.
}
} end.

We label by *iteration* the action of solving an optimal control subproblem $P_{k,N}$ by Algorithm 3.2, either at the initialization phase or through its *while* loop. Observe that Algorithm 3.2 requires N iterations. We define the *complexity* of the algorithm to be the number of convex programs that are solved by Algorithm 3.1 throughout all the iterations of Algorithm 3.2. Let β_i denote the number of blocks with which Algorithm 3.2 enters its *ith* iteration, including $\bar{B}_0$ in Algorithm

3.1. Furthermore, let τ_i denote the number of convex programs Algorithm 3.1 solves during the *ith* iteration of Algorithm 3.2. To illustrate, observe that the first iteration involves $P_{N,N}$ at the initialization phase, and therefore $\beta_1 = 1$ and $\tau_1 = 1$. Now consider the second iteration and the action of Algorithm 3.1. There are two blocks to be considered, namely $\bar{B}_0 = \{C_{N-1}\}$ and $\bar{B}_1 = \{C_N\}$, and hence $\beta_2 = 2$. Now suppose that Algorithm 3.2 exits after solving $Q_{N-1,N-1}$. In this case $\tau_2 = 1$. Moreover, at that point of exit there are two blocks, each consisting of the respective jobs $\{C_{N-1}\}$ and $\{C_N\}$. Entering the third iteration there will be the additional block consisting of the job $\{C_{N-2}\}$, and therefore, the blocks considered by this iteration will be $\bar{B}_0 = \{C_{N-2}\}$, $\bar{B}_1 = \{C_{N-1}\}$, and$\bar{B}_2 = \{C_N\}$, and therefore, $\beta_3 = 3$. On the other hand, if at the second iteration Algorithm 3.1 exits after solving $Q_{N-1,N}$, then $\tau_2 = 2$. In this case the next iteration will involve only two blocks, $\bar{B}_0 = \{C_{N_2}\}$ and $\bar{B}_1 = \{C_{N-1}, C_N\}$, and hence $\beta_3 = 2$.

Recall that there are exactly N iterations, and that the first block in the *ith* iteration consists of the single job $\{C_{N-i+1}\}$. Moreover, since Algorithm 3.1 concatenates blocks, we have that $\tau_i \leq \beta_i$.

Denote the complexity of Algorithm 3.2 by $\Gamma(N)$, and observe that

$$\Gamma(N) = \sum_{i=1}^{N} \tau_i. \tag{3.7}$$

Since the *ith* iteration may involve as many as i blocks, it appears that $\Gamma(N)$ may be of the order-of-magnitude of N^2. However, the following result gives a sharper bound.

Theorem 3.2. The complexity of Algorithm 3.2 is bounded from above by $2N - 1$.

4 Conclusions

This paper has presented a framework for optimal control of hybrid dynamical systems. A single-stage manufacturing system has been investigated, and its analysis revealed a number of structural properties, resulting in strong optimality conditions. A low-complexity algorithm has been developed. Future research will focus on applications of the analysis and algorithms developed here to optimal control of large-scale enterprise systems. Of a special interest is the question of whether the optimal control sequence can be computable via state feedback, as it has been done in [7] for a related problem.

References

[1] M. Athans and P. Falb, *Optimal Control*, Mc.Graw Hill, New York, NY, 1966.

[2] M.S. Branicky, V.S. Borkar, and S.K. Mitter, "A Unified Framework for Hybrid Control: Model and Optimal Control Theory", *IEEE Trans. on Automatic Control*, Vol. 43, pp. 31-45, 1998.

[3] A.E. Bryson and Y.C. Ho, *Applied Optimal Control*, Hemisphere, New York, NY, 1975.

[4] C.G. Cassandras and D.L. Pepyne, "Optimal Control of a Class of Hybrid Systems", in *Proc. 36th IEEE Conf. on Decision and Control*, pp. 133-138, San Diego, California, December 10-12, 1997.

[5] C.G. Cassandras, D.L. Pepyne, and Y. Wardi, "Optimal Control of Hybrid System Models of Manufacturing Systems", *Manuscript*, Dept. of Manufacturing Engineering, Boston University, 1999.

[6] A. Deshpande and P. Varaiya, "Viable Control of Hybrid Systems", in *Proc. 35th IEEE Conf. on Decision and Control*, pp. 1196-1201, 1996.

[7] M. Gazarik and Y. Wardi, "Optimal release Times in a Single Server: An Optimal Control Perspective", *IEEE Trans. on Automatic Control*, Vol. 43, pp. 998-1002, 1998.

[8] J. Lygeros, J. Tomlin, and S. Sastry, "Controllers for Reachability Specifications for Hybrid Systems", *Automatica*, Vol. 35, pp. 349-370, 1999.

[9] D.L. Pepyne and C.G. Cassandras, "A Calculus of Variations Approach to the Optimal Control of Discrete Event Dynamic Systems", in *Proc. 36th IEEE Conf. on Decision and Control*, pp. 3550-3555, San Diego, California, December 10-12, 1997.

[10] D.L. Pepyne and C.G. Cassandras, "Modeling, Analysis and Optimal Control of a Class of Hybrid Systems", *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 8, pp. 175-201, 1998.

[11] Y. Wardi, C.G. Cassandras, and D.L. Pepyne, "Algorithm for Computing Optimal Controls for Single-stage Hybrid Dynamical Systems", Manuscript, Georgia Institute of Technology, School of ECE, July 1999.

Automation of Command and Control Planning Using a Markov Process-Based Technique

Stephen Stubberud, Daniel Damouth,
Peter J. Shea, Charlene Kowalski, Allen Ott
ORINCON Corporation
E-mail: stubberu@orincon.com

Allen Stubberud
University of California Irvine

Abstract

The development of a closed-loop feedback control technique for a command and control planning problem is presented. This new method, which combines an adaptive Markov control with a procedural reasoning system, opens the possibilities of an automated methodology to develop plans for military campaigns. In this paper, we outline the overall approach for developing a hybrid controller to control route selection for air mission planning. We apply the technique to a simple example that obviates many of the future concerns and issues of the closed-loop control to the command and control problem.

1. Introduction

"The general who wins makes many calculations before the battle is fought. The general who loses a battle makes but few calculations beforehand. Thus, do many calculations lead to victory, and few calculations to defeat; how much more no calculation at all!"

Sun Tzu wrote these words in his tome *The Art of War* [1] more than 2,500 years ago. Interestingly, these words still ring true for military planning today. Military plans provide practical guidance for the employment of forces at the operational level of war. In today's environment, the number of contingencies and variations that can occur has increased tremendously. Automation was therefore introduced into the various planning components to facilitate the planning process.

Unfortunately, these planning components were developed to perform optimization without consideration of the coupling between the dynamics of the other parts of the planning process. In the past, the planning components consisted of many subcomponents, i.e., a router, scheduler, and allocator, that perform optimization without considering the goals of the other processes. These component processes typically operate in serial with the output of one component passing information to the next component. This scheme has led to many problems in the planning community because a minor change in one component's parameter set could cause drastic changes in the outputs of other planning components. Besides this sensitivity issue, the order of the planning components in the processing chain causes concern. A given component may require information from several other planning components. However, one of these planning components may require partial information from the given planning component. By not considering the interaction, a processing breakdown could occur or solutions could be based on highly uncertain estimates. Finally, even with automation, parameter changes and changes to the inputs still require operators to adjust the parameters in an effort to achieve closer to optimal results.

While automation of the individual planning components brought improvements to the planning process, the lack of feedback control prevented the automation of the process as a whole. In this paper, we introduce a closed-loop control scheme to improve the planning process. Feedback control will provide

- Improved results with respect to the long-range goals
- Rapid reaction to changes in the data
- Processing without delays due to partial information from observations and other planning processes
- Reduction in sensitivity to uncertainty and lack of assessment information
- Reduction in the required number of operators.

The technique presented here is a hybrid Markov and Procedural Reasoning System (PRS) [2] control system. Using this control scheme, we have multiple feedback loops. These loops originate from the plant and its

sensors, the planning components, and the operator. Each loop provides a different set of measurements for changing both the control input and the controller parameters (adaptation).

The plant that we are affecting through the development of mission plans is the campaign world. Each plan provides input to drive the state of the world to a new state, preferably a desired state. While our actions have repercussions on the overall state of the world, we only have strong reachability and observability over a small subset of this world. Our actions can have direct and indirect effects on states of the war, which in turn affect the states of our campaign. There are also external events, over which we have no control, that can have both beneficial and deleterious effects on our states. We look at these effects as disturbance inputs.

The estimate of the campaign state is provided through observations and assessments along with direct reports. Unlike many existing planning systems, we consider the planning components to be the input-coupling functions of our plant. They process the control input and provide a plan to be executed in the campaign. They also provide feedback to the controller to make adjustments to improve performance before execution.

The hybrid controller provides input to the planning process. In this paper, we develop this control technique to augment the route planning component of the planning process in order to improve performance in the campaign.

We begin the paper with an overview of the current planning methodology. We then discuss routing and its implications on the planning process. Then we develop the control methodology. This is followed by a simple routing example. We conclude by providing a roadmap of the future direction of the control through the planning process and some of the implementation issues that we foresee along with the proposed solutions.

2. Current Planning System Implementation

Current state-of-the-art tactical planning systems are able to process less than a globally optimal execution order. In these systems, optimality is handled by partitioning the problem into either processes or logical clusters. Process partitioning is a method that limits the combinatorial problems of global optimality by stepwise refining the plan generation process. For example, an assignment or allocation technique might be used to bound the routing and scheduling solution. Then assignment changes are made only when schedule or route calculations trigger exceptions such as feasibility or survival thresholds. In contrast, logical partitioning is a method that bounds the combinatorial problems of global optimality by limiting the sphere of control of the calculation. For example, a single strike package might be assigned, routed, and scheduled independent of the rest of the campaign. Optimality is only considered across the package with variations provided by external modification of the initial conditions.

In both of these approaches and in current systems in general, global optimality is considered as constraint processing or managed by human operators. These operators must either intuitively or empirically (using multiple trials) understand the relationship between constraints and the calculated results. For example, in the TRICOMS Strategic Planning environment, current techniques were shown to require months in operational configuration, e.g., Single Integrated Operation Plan (SIOP) updates, and at best days using the best experimental technologies, e.g., Strategic Adaptive Planning Experiment (SAPE). Our approach provides a technology revolution that will benefit tactical command and control by providing cross-domain planning as well as response times on the order of minutes.

3. Define the Router Problem

For our initial application of feedback control, we are focusing on aircraft route planning. In simple terms, aircraft route planning is the task of figuring out the path that aircraft should take to get to get where they need to be to perform their mission. In the chain of events, this planning occurs after targets are assigned to units and after aircraft types are assigned to targets. This route planning applies to multiple campaign planning levels: that is, when devising routes for a small set of aircraft with common objectives (i.e., raid level) and for individual sorties (i.e., sortie level).

Current automated route planners typically perform point-to-point route planning for only the mission at hand. That is, given a set of inputs and domain knowledge, they generate multiple route options and evaluate the route segments to find routes that are feasible, optimize resources, and minimize friendly losses. Figure 1 illustrates the inputs, evaluation, and outputs of these route planners.

Although these route planners evaluate complex, often conflicting factors to find practicable routes, they suffer from two important deficiencies. First, they fall into the category of open-loop systems. That is, given a set of inputs, they produce an output for the mission at hand and pass it to the next component in the processing

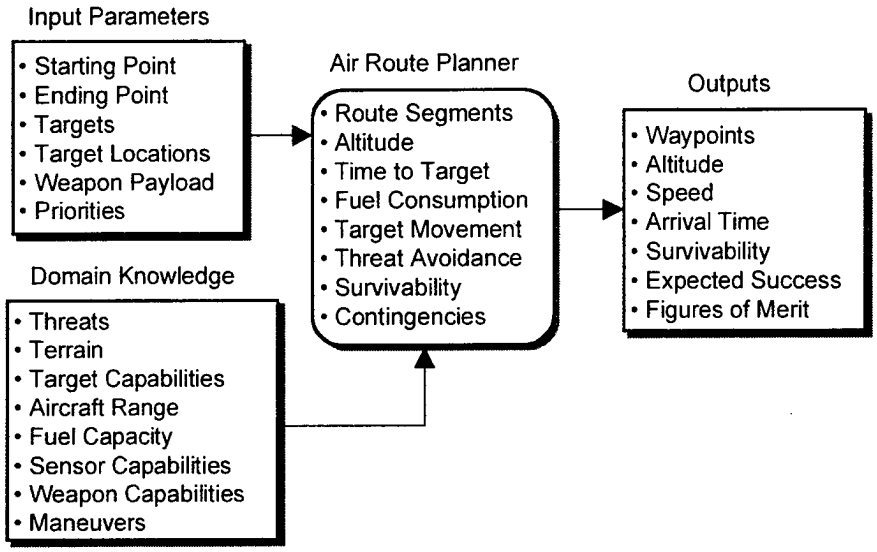


Figure 1. Automated planners consider many complex factors to find the best routes for aircraft

chain. However, in the real world, dynamic events such as attrition and priority changes cause changes in planner inputs, which in turn impact the planner output. To accommodate these changes, current planners must restart the route planning process from the beginning, losing any intermediate results that may be of value.

Furthermore, these route planners perform single mission optimization. That is, they neglect to evaluate the merit of routes for the overall campaign, which includes multiple raids over a series of days. For example, the route planner may recommend a route for the current mission that expends resources having a higher benefit to the campaign if deferred for use in a subsequent mission.

By introducing feedback control into the route planning process, we address these two deficiencies. In our concept, we depict campaigns as a network of force-level planning (FLP) events that are connected by raid-level planning (RLP) events. In this approach, FLP events can be thought of as potential states of a campaign, while RLP events are the transitions between states. Figure 2 illustrates the concept. At any FLP state, there is a group of possible next FLP states and a preferred RLP route plan to reach each of those states. In overall campaign planning, we use automated route planners to generate routes for each of the transitions and evaluate the routes to measure overall feasibility, quality, and expected success for the campaign. To keep the calculations manageable, we aggregate related states together. This approach incorporates feedback and achieves multiple mission optimization for the entire campaign.

Furthermore, as raids or other dynamic events occur, updates to planning inputs (e.g., priority changes, target damage, aircraft status) become feedback inputs to the next FLP state. Since we already have intermediate results from the planning stage, we can both expedite calculations to achieve a fast response and perform more in-depth calculations to achieve higher quality results. Additionally, we can minimize processing resources by pruning the state transitions that can no longer occur. In the figure, nodes outside the boxes are pruned. This approach incorporates feedback, takes advantage of intermediate results, and integrates the planning and execution stages.

4. Markov/PRS Feedback Control For Planning Systems

Unlike the open-loop planning systems, where the planning components are the systems being controlled, the closed-loop system uses the planning components as the input coupling function to the plant. In the open-loop control, the planning process is the plant. The output is a plan. For our closed-loop approach, the plant is the campaign, whose states are measured through reports, assessments, and observations. The plans that we develop are executed and then monitored to determine how our control affected the dynamics of the campaign.

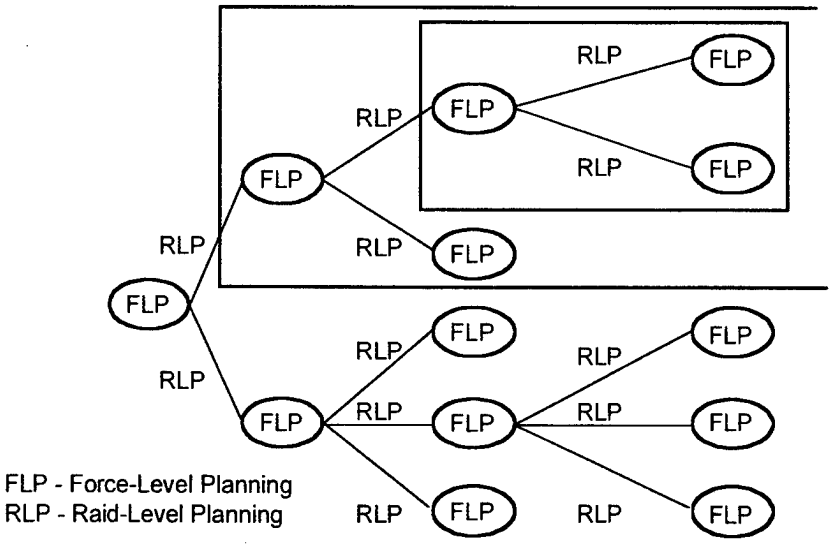


Figure 2. FLP states and RLP transitions facilitate the evaluation of route plans in terms of the overall campaign

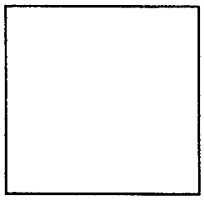
The dynamics of the campaign are highly complex. A stochastic model would be required to emulate these dynamics and develop a control law. Also, the inputs to the planning components and the inputs from the operators are context-based. On the other hand, the control law is based on a real number representation. The operator inputs and the inputs to the planning components are based on text information and specific input ranges. For these reasons, we approached the control system with the hybrid solution of a Markov control and a PRS control agent. The Markov controller provides a modeling technique for the states of interest while the PRS system provides the interpretation between the control law and the external components.

4.1. Adaptive Markov Control

We initiate the development of our control law with a Markov chain that incorporates rewards [3]. We consider the problem as a Markov chain with N states that receives a reward for transitioning from state i to state j, where the value of the reward is a real number r_{ij}. Note that this reward could be negative, indicating a loss or penalty for a particular transition. Let $V_i(n)$ be the expected (in the probabilistic sense) total earnings in the next n transitions if the system is now in state i. This can be written in the form of a recursion relationship as

$$V_i(n) = \sum_{j=1}^{N} p_{ij}[r_{ij} + V_j(n-1)] = \sum_{j=1}^{N} p_{ij} r_{ij} + \sum_{j=1}^{N} p_{ij} V_j(n-1), \quad i = 1,2,\cdots,N \quad n = 1,2,\cdots,M.$$

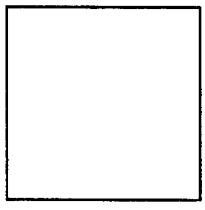
The first term is the expected reward the system will receive in the transition from time n-1 to time n and the second term is the expected reward for the first n-1 transitions starting in state i. If we define the first term as



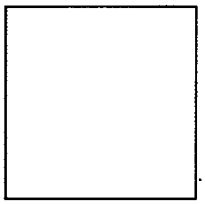
then the recursive relationship can be written

$$V_i(n) = \sum_{j=1}^{N} p_{ij} r_{ij} + \sum_{j=1}^{N} p_{ij} V_j(n-1) = q_i + \sum_{j=1}^{N} p_{ij} V_j(n-1), \quad i = 1,2,\cdots,N \quad n = 1,2,\cdots,M.$$

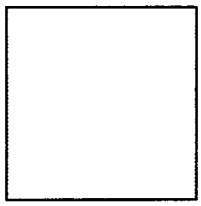
To extend this to the Markov decision process, we suppose that at each time point there are several possible transition probability matrix choices with corresponding rewards. For example, suppose that for a system with N states there are K possible alternatives of transition probabilities. Then we define the probabilities of the k-th alternative as



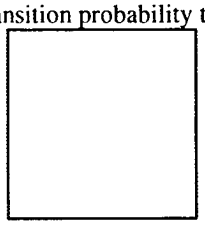
where each transition probability has its own reward structure given by



We define



and assume that we wish to optimize our expected reward starting from the initial state and moving through a total of n time steps. The (dynamic programming) equation, known as Bellman's equation, that solves this problem to determine which alternative transition probability to choose at each time step is



(1)

The adaptation of the Markov controller comes from the ability of the controller to adjust the transition probabilities through action selection.

4.2. Markov Control As Applied to Route Planning

We approached route planning in a simple manner to determine if the Markov control approach has merit. The first step in developing a Markov controller is to define the time or event step between each set of states. For the routing problem, we initially define the event steps as missions. Our missions are defined as all flights flying sorties against their defined targets and setting up for the next mission.

From this point, we develop the state space. To define the components of a state, we use the rule that each member of the set must be mutually exclusive and collectively exhaustive for each element of the index set. where the index indicates a particular value for the state vector. While this definition could be done "on-line" as conditions change during the course of the n-step process, we created a static set of states. These states represent an aggregated subset of the states of the plant. The states' aggregation varies according to how far in the future the state exists.

The current state and the one-step predicted state are defined as

$$\begin{matrix} \text{Target}_1 \\ \vdots \\ \text{Target}_m \\ \text{Aircraft}_1 \\ \vdots \\ \text{Aircraft}_n \end{matrix}$$

where the elements of the state is a 1 if the target/aircraft is destroyed/unavailable and a 0 if the target is operational/available. As we predict further ahead in time, we begin to aggregate the states. The aggregation serves three purposes. First, it incorporates modeling and measurement uncertainty into the state itself. Second, it increases the development of the transition probabilities over a longer period of time. Finally, the state aggregation reduces the number of possible states into more manageable sets.

For the simple case provided in this paper, our first state aggregation becomes

$$\begin{matrix} \text{Number of Targets Remaining} \\ \text{Number of Aircraft Available} \end{matrix}$$

Each of the elements of this new state vector is a positive integer. The aggregation of the states continues into more generic representations. For example, the set of our long-term state vectors is defined as

All Targets Operational/*All* Aircraft Available

⋮

No Targets Operational/*All* Aircraft Available

⋮

All Targets Operational/*No* Aircraft Available

⋮

No Targets Operational/No Aircraft Available

Depending on the granularity desired, the set may include such states as *Most*, *Many*, and *Some*.

The long-term view is that we desire all of the targets to be destroyed with minimal friendly casualties. Specific aircraft losses and specific surviving targets would not be of importance. However, in short- to medium-term planning, specific groups of aircraft and specific targets would be of more importance. As can be seen in our long-term state representation, we can

generalize the destruction of targets and aircraft. Also, the reduction in the number of states clearly makes the development of a solution set tenable in that, for the development of a Markov controller, we must estimate the transition probabilities based on statistics or whatever experience is available for this type of process. Even without uncertainty, the size of the problem for more than a handful of missions in a true campaign would lead to combinatoric explosion.

While the definition of the states is of major importance, the key issue of using the Markov process lies in how we define our probabilities for transition. While the control process is used to modify the transition probabilities, the effects of the control will more keenly be seen in the short term. Long-term predictive transitions can be defined and modified using a rule base. We develop a series of transitions loosely developed on the probabilities of kill and survival for attacking various targets. As more deadly targets are destroyed, survivability would increase. Changes in the probability would be similarly adjusted based on the capabilities of the available aircraft. For our example, each target has a fixed capability and the aircraft are identical. The aggregated states and associated transitions are defined using Bayes' rule with the products of the probabilities of kill and survival and their complements. For example, given two targets and one aircraft, the initial state could transition from [*Aircraft, Target#1, Target#2*] = [0,0,0] to [0,0,0], [0,0,1], [0,1,0], [0,1,1], [1,0,0], [1,0,1], [1,1,0], and [1,1,1] in one event. The associated probabilities, given equal probabilities of assigning the aircraft to target one or target two, would be

$$[0,0,0] \blacklozenge [0,0,0] : P_S(1-0.5P_{K1}-0.5P_{K2})$$
$$[0,0,0] \blacklozenge [0,0,1] : 0.5P_S P_{K1}$$
$$[0,0,0] \blacklozenge [0,1,0] : 0.5P_S P_{K2}$$
$$[0,0,0] \blacklozenge [0,1,1] : 0$$
$$[0,0,0] \blacklozenge [1,0,0] : (1-P_S)(1-0.5P_{K1}-0.5P_{K2})$$
$$[0,0,0] \blacklozenge [1,0,1] : 0.5(1-P_S)P_{K1}$$
$$[0,0,0] \blacklozenge [1,1,0] : 0.5(1-P_S)P_{K2}$$
$$[0,0,0] \blacklozenge [1,1,1] : 0$$

respectively, where P_S is the probability of survival of the aircraft and P_K represents the probability of killing target i. The selection of the route is based on the reward. The cost or benefit of the transition, described in Equation (1), can vary by changing the value of particular targets or aircraft. This variation can be adjusted-based on the context or strategy of the mission. To provide contextual and human variations to the control, we incorporate a procedural reasoning agent-based system to wrap around the Markov control law.

4.3. Procedural Reasoning Control

We envision using an agent-based controlling scheme to implement signals from the Markov controller. While these signals may be in the form of real numbers (e.g., desired probability of kill or probability of survival, time criticality, etc.), the control signals required to drive route planning components are symbolic process invocations that have many discrete parameter set choices and ordering constraints. We defined the Agent Technology for Agile Control (ATAC) component to intelligently translate the master control vector into the proper planning component procedure call. ATAC is based on PRS, which uses the notion of rational *belief-desire-intention* (BDI) agents that can reason in complex ways (using *knowledge areas*) about dynamic processes, while ensuring appropriate responsiveness and control. ATAC functions as a hierarchical reactive system to control and execute procedures for managing the route planning components in response to the master control process.

4.4. The ATAC Architecture

As shown in Figure 3, the ATAC control agent accepts input from the controller, from sensors and evaluators (in the PRS world model), and from operators (e.g., objectives and guidance). ATAC maintains a library of control plans (which are constructed using an off-line control plan editor) that are represented as PRS knowledge areas. Control plans, which are procedures for controlling the route planning components, consist of sequences of actions, each of which may be a primitive action invoking a subsystem or a subgoal assertion that will lead to nesting of plans. Each control plan is associated with a goal that it attempts to achieve, an operating context in which it is routing stealth planes, with the context that it must generate the route in under one hour, and the constraint of flying at a certain altitude.

Priorities are monitored continuously; if an appropriate plan suddenly becomes of higher priority than a currently executing plan, the system will switch to the higher priority plan. Priority changes may occur due to a change of goals or the introduction of a new goal, or by new context information. This allows the fluid integration of goal- and event-driven activity.

The ATAC control agent matches the current operating context (which includes information from the world model and constraints imposed by the master control vector) and high-level system goals — such as military objectives — to control plans. The result is a set of control plans that are applicable to the current context and are appropriate for the goals.

The flexibility of the ATAC architecture also facilitates control over tradeoffs such as time criticality versus solution optimality. For example, if time criticality is high — a route plan must be developed within a very short amount of time — a control plan problem at once, even though they would normally be run in sequence to allow iterative analysis of their outputs. This parallel execution results in the faster creation of a plan, but at the potential cost of it being a might invoke several planning components on the less efficient plan. The converse situation (low time criticality) triggers a different control plan that executes planning tools sequentially, feeding the exact outputs of one tool into the next to achieve a more optimal solution. With a properly designed control plan library,

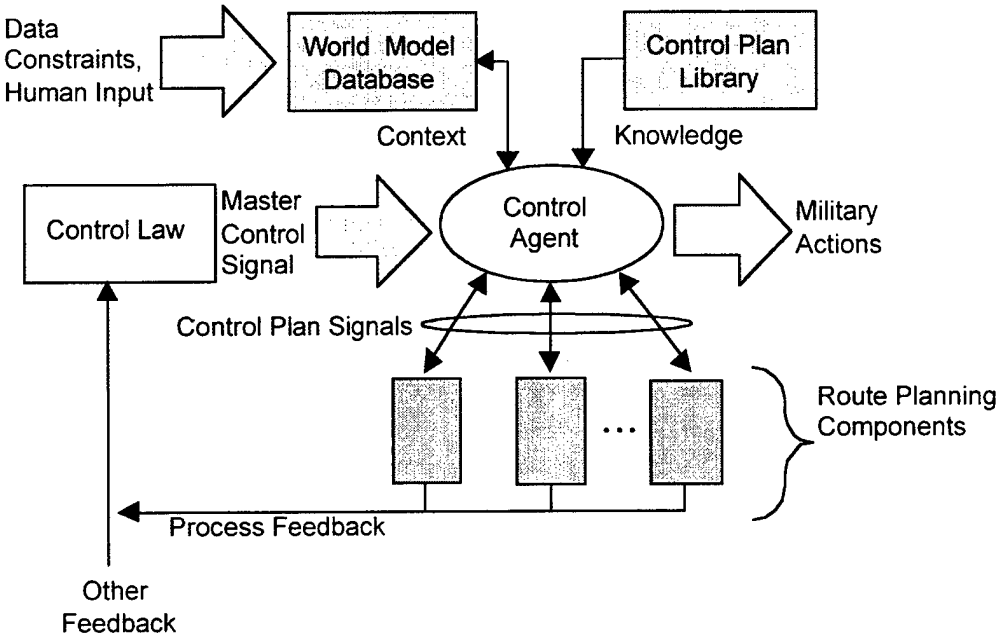


Figure 3. The control agent provides a way to incorporate the numerical controller into a contextual world

ATAC will be superbly responsive to such contextual constraints, under the guidance of the control law.

5. Simple Router Example

The scenario that we developed for the router is the suppression of enemy air defenses (SEAD). To demonstrate the feasibility of the control design, we started with one flight of two planes flying against one target. The target has the capability to defend itself.

We limit the aircraft approach to the target to three separate altitudes. Each altitude changes the probability of survival and the probability of kill. The loss of each aircraft has a cost and the destruction of the target has a benefit. Our goal is to destroy the target with minimal losses. From this we create the cost function

$$\begin{aligned} \text{maximize } & P_K b_1 + (1 - P_{S1})c_1 + (1 - P_{S2})c_2 \\ \text{subject to } & 0 \le P_K \le 1 \\ & 0 \le P_{S1} \le 1 \\ & 0 \le P_{S2} \le 1 \\ & b_1 \ ?\ 0 \\ & c_1 \le 0 \\ & c_2 \le 0 \end{aligned}$$

where b indicates the benefit of destroying the target and c indicates the cost of losing an aircraft. The cost function's incorporation of the probabilities allows us to estimate the best route. The probabilities of survival and probability of kill, $[P_S, P_K]$, are defined as

[0.8,0.1] — *high*
[0.7,0.3] — *medium*
[0.5,0.6] — *low*

at three different altitudes. The low-altitude transition matrix for the eight states described by [*aircraft1*,*aircraft2*,*target*] to the future-time same states is

$$P_{low} = \begin{matrix} 0.10 & 0.15 & 0.10 & 0.15 & 0.10 & 0.15 & 0.10 & 0.15 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.20 & 0.30 & 0 & 0 & 0.20 & 0.30 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.20 & 0.30 & 0.20 & 0.30 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}$$

Figure 4 shows the limiting probability of achieving the particular states in the Markov process if we repeatedly applied the same strategy. Some of the states are unstable or are transition states that have a probability zero of occurring in the large.

Assuming a benefit of $b_1 = 1$ for the target's destruction and a cost of $c_1 = c_2 = -1$ for an aircraft's destruction, the selected route would be at altitude *low*. As shown in Figure 5, the best-scoring approach would be Strategy 3. The probability of killing the target with at least one surviving aircraft is approximately 0.6. The probability of killing the target though is greater than 0.8. On the other hand, if we change cost and benefits to desire a higher safety factor for the plan, $c_1 = c_2 = -3$ for the destruction of an aircraft, the selected strategy would be a *medium* altitude approach as shown in Figure 6. The probability of killing the target with at least one aircraft surviving is again approximately 0.6, but the probability of both aircraft being destroyed is approximately 0.4, which is less than Strategy 3's value 0.43.

6. Future

The results from this paper lay the groundwork for implementing the Markov/PRS control methodology in a command and control structure. The design outlined in this paper has the potential to provide improved capabilities in a command and control process.

While this development is still in the embryonic stages, we foresee many challenges and areas of research that will arise, including investigation of the effects of aggregation, use of fuzzy logic to perform the aggregation, provision of a larger state space (scenario), incorporation of uncertainty into the measurements from the plant, and incorporation of a scheduler and allocator into the planning system.

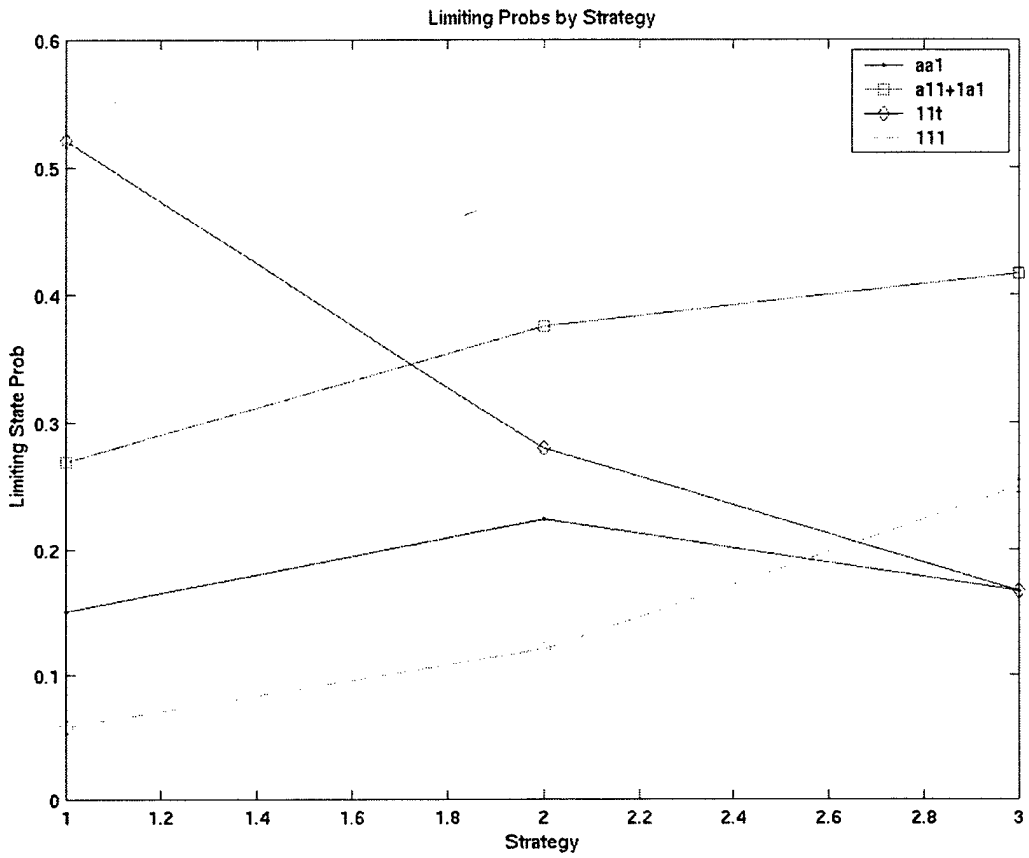


Figure 4. Different altitude approaches have different limiting probabilities on the possible states

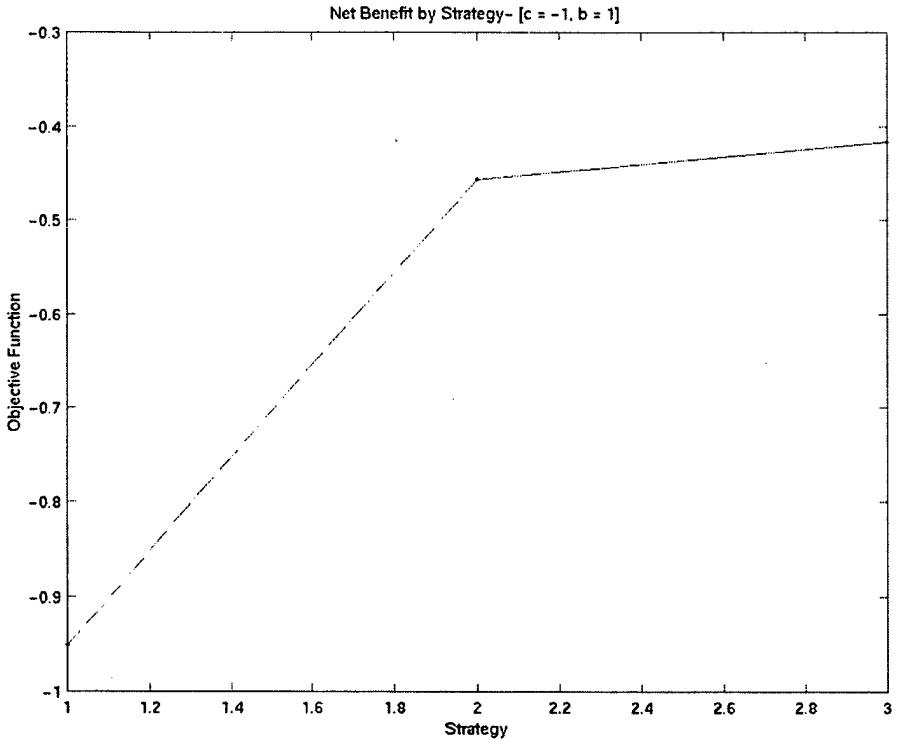


Figure 5. With equal weighting on cost and benefits, a low-altitude attack is preferred

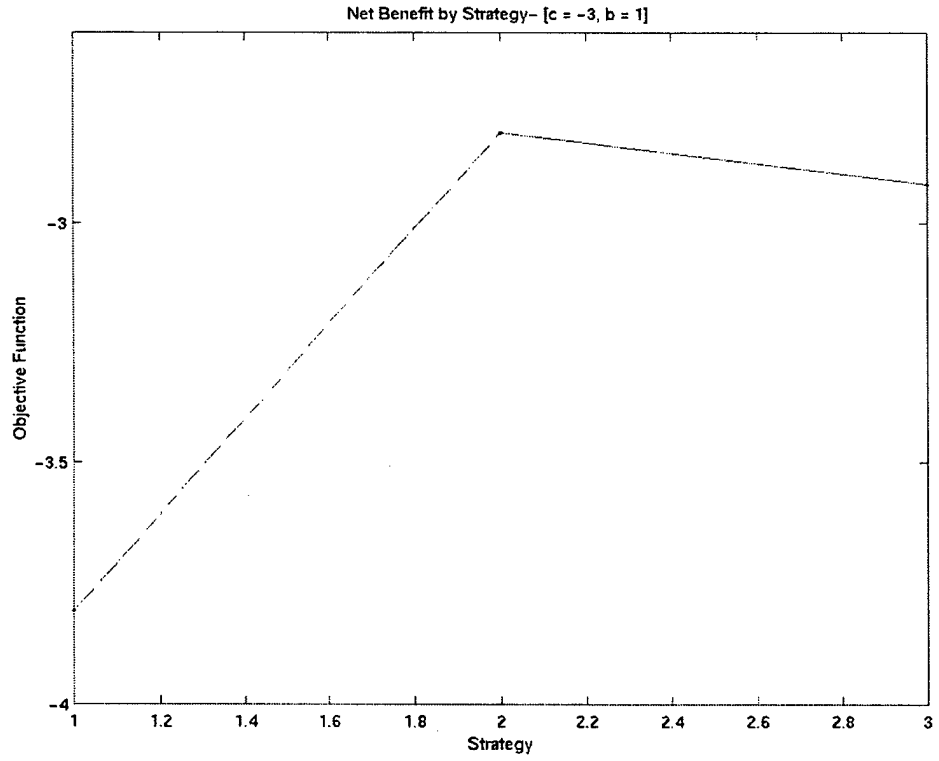


Figure 6. With the priority to save the aircraft, Strategy 2 becomes the preferred strategy

In [4], the importance of aggregation was discussed in applying Markov control to traffic control. The concern was that the number of states could grow quickly. This was also discussed in the Markov control section above. The number of control variations may become too big to handle. In our state representation, the number of possible states grows as $2^{\#targets+\#aircraft}$. In a moderate-sized campaign, this number becomes quite large and the interpretation of the problem becomes difficult. However, such fine granularity may not be required. We may not need to know a tail number, but just the class of aircraft or region to which it belongs. Each variation in aggregation may have a deleterious or a beneficial effect. Only analysis and experimentation will provide the results.

One aspect of aggregation is the mapping of values to linguistically defined states. Instead of a number, the state may be defined as *lots, some,* or *few*. The mapping from values to linguistics could possibly be fuzzy and not crisp. While the fuzzy mapping is trivial (see [5]), the incorporation of the fuzzy membership into the Markov controller will require a modification in the approach. Fortunately, Watkins [6] has made a generalization of Bayes' theorem as applied to the Kalman filter, which is a Markov process. We plan to extend this work into our Markov control law.

Once we provide these aggregation capabilities and experimentation, we can more readily investigate the approach with larger and more realistic scenarios. As our scenarios become more realistic, we will investigate the use of uncertainty in the measurement reports from the plant. At times we will be uncertain of our battle damage assessment and the availability of the aircraft after a mission. Uncertainty will provide us with the ability to analyze the sensitivity of our system to noise in the closed loop as compared to that of the current open loop.

While more subproblems will become apparent, our current challenge is to incorporate other planning components into the system. With the addition of a component such as an allocator or scheduler, we can begin to work on the input coupling and sequencing problem, which is the crux of the command and control problem.

7. References

[1] Tzu, S., *The Art of War*, J.C. Clavell (ed.), Dell Publishing, New York, 1983.

[2] F. Ingrand, M. Georgeff, and A. Roa, "An Architecture for Real-Time Reasoning and System Control," *IEEE Expert*, 7(6), December, 1992, pp. 34-44.

[3] *The Control Handbook,* W.S. Levine (ed.), CRC Press, Boca Raton, Florida, 1996.

[4] X.-H. Yu, and A.R. Stubberud, "Markovian Decision Control For Traffic Signal Systems," *Proceedings of the 36h Conference on Decision and Control*, San Diego, CA, 1997, pp. 4782-4787.

[5] Kosko, B., *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[6] Watkins, F.A., *Fuzzy Engineering*, Ph.D. thesis, University of California Irvine, Department of Electrical and Computer Engineering, June, 1994.

Section 4

Distributed and Agent-Based Strategies

Synthetic Pheromones for Distributed Motion Control

H. Van Dyke Parunak
ERIM Center for Electronic Commerce
PO Box 134001
Ann Arbor, MI 48113-4001 USA
vparunak@erim.org

Sven Brueckner
DaimlerChrysler Research and Technololgy
Alt Moabit 96A
D-10559 Berlin GERMANY
sven.brueckner@daimlerchrysler.com

Abstract

A common control problem in applied domains such as military operations, manufacturing, and logistics is to change the locations of entities under certain constraints. For example, in a military air combat application, fighter jets must move in concert with their support facilities (such as tankers and AWACS). The centralized mechanisms traditionally used to address these problems can easily become bottlenecks and points of vulnerability. Multi-agent negotiation schemes enable the entities to maintain the desired constraints among themselves in decentralized fashion, but may be complex to implement and manage. A particularly simple form of coordination, inspired by pheromone mechanisms in natural insect populations, replaces central coordination and agent-to-agent communication with interaction through a shared environment. We illustrate this mechanism and some of the associated design issues using the classical Missionaries and Cannibals movement problem.

1. Introduction

Many applied problems in domains such as military operations, manufacturing, and logistics require that individual entities move from one location to another while maintaining certain constraints. For example,

- resources must be allocated to targets in a way that achieves tactical objectives;
- parts must move through a factory in a way that preserves process order and balances machine load;
- shipments must be routed to provide timely delivery within capacity constraints of a transportation network.

These problems have traditionally been addressed with centralized planning and control mechanisms. In highly dynamic domains such as combat management, the time required for central coordination may not permit timely response to rapidly changing threats, and the site at which central control takes place is a vulnerability that can place the entire operation at risk.

Recent research in multi-agent systems has developed negotiation schemes that enable the entities to maintain such constraints. These mechanisms effectively decentralize system control, but can require sophisticated inter-agent communication and significant processing within individual agents. There are several motives for seeking simpler mechanisms.

- Requirements for conventional negotiation restrict the low-level deployment of these techniques. It is unlikely that one would equip each box of field rations or each seat assembly en route from supplier to final assembly with the resources to perform such negotiation.
- The dynamics of these mechanisms are poorly understood, and in some cases may become intractable, leading to inadequate performance.
- In many realistic settings, the complexity of designing interlocking protocols so that they all work correctly is non-trivial.
- In a military context, high semantic content in inter-agent messages is a point of vulnerability to evesdropping.

Insect colonies perform sophisticated motion coordination and control using neither central coordination nor direct agent-to-agent communication. Instead, each agent deposits chemical marker or "pheromones" in the environment. The environment aggregates and evaporates these deposits over time.

We have been exploring the application of analogous mechanisms to engineered systems. The pheromone mechanism enables up-to-date many-to-many information sharing across agents in a completely distributed manner, without direct coupling of agents. The active environment that supports aggregation can be analyzed using formal models, and important parameters (e.g., time until information is available, error in approximation) can be computed for each scenario [3].

This indirect approach has many benefits: it decouples agents, off-loads computation to a server (the place) without compromising agent performance or autonomy, provides "forgetting" over time, and generates a flow field that can support agent coordination. However, potential users need guidelines on how to design such agents and how their performance varies with the pheromone vocabulary available to them

This paper illustrates the basic principles and their performance on a classical problem in motion planning, the Missionaries and Cannibals (MC) problem. Section 2 provides background by introducing the MC problem, summarizing the basic mechanisms inspired by insect pheromones, and reviewing related research literature. Section 3 outlines the algorithms we have implemented and discusses ways to measure their effectiveness. Section 4 discusses some simple experiments applying these mechanisms to MC. Section 5 summarizes our results and outlines ongoing directions for our research.

2. Background

This paper illustrates the potential of synthetic pheromones using the well-known Missionaries and Cannibals problem, building on previous work using similar mechanisms.

2.1. The Missionaries and Cannibals Problem

Three missionaries and three cannibals find themselves together on one bank of a raging river, with a dugout canoe capable of carrying only one or two people. If at any time the cannibals outnumber the missionaries on either bank of the river, the cannibals will eat the missionaries. The problem is to plan a sequence of moves that gets all six people safely across the river.

The problem has thirty-two states (0-3 cannibals * 0-3 missionaries on the left bank, * 2 possible locations for the boat), but only twenty states are safe (that is, avoid a preponderance of cannibals over missionaries). We denote a state by a three-tuple <BoatBank, LeftCannibalCount, LeftMissionaryCount>. Figure 1 shows the live states (numbered for ease of reference) and the accessibility relations among them.

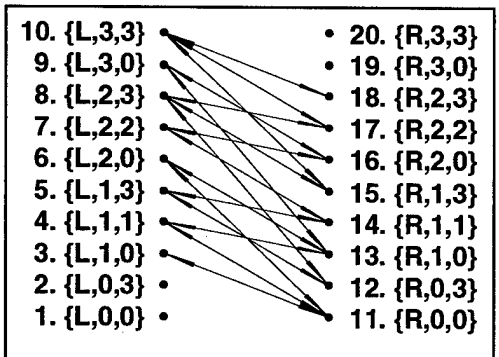


Figure 1: Safe States and Transitions

Since the boat must alternate between banks, the allowed transitions form a bipartite graph. There are two shortest paths through this graph from state 10 {L, 3, 3} to state 11 {R, 0, 0}, each eleven steps long, and differing only in the penultimate state: {10, 15, 8, 12, 5, 14, 7, 16, 9, 13, 4, 11} and {10, 15, 8, 12, 5, 14, 7, 16, 9, 13, 6, 11}. Longer solutions also exist, containing cycles that revisit certain states. The problem is fairly easy to solve centrally. Our challenge is to define behaviors for the individual missionaries and cannibals such that they find their way across the river without central control.

Realistically, ant-like agents or even decentralized methods are not appropriate for toy problems with a state space as restricted as this one, particularly when the answer is already well known. However, this exercise is useful from two perspectives. First, the problem is well known in the AI community, so it is useful as a tutorial introduction to pheromone methods. Second, the problem is complex enough to reveal important characteristics of the mechanisms of interest and help us develop techniques for engineering such agents. In this paper, it enables us to study the effect of giving our agents different amounts of knowledge within the constraints of the pheromone approach.

2.2. Insect Examples

Insects perform impressive feats of coordination without direct inter-agent coordination, by depositing pheromones in the environment and then sensing them [12]. For example, ants construct networks of paths that connect their nests with available food sources. Mathematically, these networks form minimum spanning trees [9], minimizing the energy ants expend in bringing food into the nest. Graph theory defines a number of algorithms for computing minimum spanning trees, but ants do not use conventional algorithms. Instead, this globally optimal structure emerges as individual ants execute the following behaviors:

1. Avoid obstacles. Whatever an ant does, it will not aimlessly push against a wall.
2. Wander randomly.
3. If the ant is not holding food, drop nest pheromone as it walks, and preferentially wander in the direction of food pheromones.
4. If the ant is holding food, drop food pheromone as it walks, and preferentially wander in the direction of nest pheromones.

5. If the ant finds itself at food and is not holding any, pick the food up.
6. If the ant finds itself at the nest and is carrying food, drop the food.

Brownian motion brings the ant arbitrarily close to every point in the plane. As long as the separation between nest and food is small compared with the ant's range, a wandering ant will find food if there is any, and a food-carrying ant will find the nest.

Only food-carrying ants drop food pheromone, and ants carry food only after picking it up at a source, so all food pheromone paths lead to food. Because only empty ants drop nest pheromone, and because all empty ants originate at the nest, all nest pheromone paths lead home. Because pheromones evaporate, paths to depleted food sources disappear, as do paths laid down by food-carrying ants that never reach home.

The initial path is not straight, but the tendency of ants to wander generates short-cuts across meanders. Overlapping pheromone paths tend to merge together into a trace that becomes straighter the more it is used. The character of the resulting network as a minimal spanning tree is not intuitively obvious from the individual behaviors, but emerges from the emulation. Similar mechanisms account for the regular structure of tropical termite mounds.

2.3. Mechanisms

We can identify two mechanisms in these behaviors that we seek to emulate: stochastic movement, and pheromone-based counting.

Stochastic search is the ultimate "weak method." By itself, it fails under combinatorial explosion. However, when tempered with other mechanisms (such as temperature in simulated annealing, or crossover in genetic algorithms, or local alignment with other agents in particle swarm optimization), it is ubiquitous in practical weak methods. It is an essential component in most models of insect behavior, and we will incorporate it in our MC model.

The real-world provides three operations on chemical pheromones that support purposive insect actions. It *aggregates* deposits from individual agents, *evaporates* them over time (with the effect of avoiding overloading and forgetting obsolete information), and *diffuses* them to nearby locations (with the effect of providing a gradient that agents can follow). Our model mimics two of these mechanisms, aggregation and evaporation. A software environment that supports these can dynamically maintain a count of the number of entities of each type that are currently present. Let $s(t)$ be the strength of the pheromone in a location at (discrete) time t, and assume that each agent deposits one unit of pheromone per unit time and that a proportion $0 \leq E < 1$ of the cumulative deposit evaporates each unit time. Then if A agents are present in a location, the pheromone strength evolves as

$$s(t+1) = E*s(t) + A.$$

This dynamic has a fixed point that can be evaluated by setting

$$s(t+1) = s(t): s(t) = A/(1-E).$$

Thus if agents move slowly compared with the time interval between deposits, each agent can estimate the population of other agents by sensing the strength of local pheromones. If different agent types (say, Missionaries and Cannibals) deposit different pheromones, the population of each species can be estimated. The strength approaches the asymptotic fixed point as a concave exponential, so that waiting times for stabilization are reasonable (about 40 steps for $E = 0.8$ and $A = 3$).

We apply these techniques to real-world problems in two ways. Sometimes actual physical entities can move immediately as indicated by pheromone-guided agents. In other cases, the real-world entity cannot tolerate this stochasticity. In such cases the mechanisms can be applied to a population of virtual agents, and the resulting emergent behavior consulted to guide the real-world entities. Because of the simplicity of these mechanisms, they can run extremely rapidly, permitting simulation faster than real time.

2.4. Comparison with Other Research

The potential of insect models for multi-agent coordination and control is receiving increasing attention. In addition to [12], a book by leading European researchers [1] outlines several mechanisms that lend themselves to practical application. Both of these reviews cover other mechanisms in addition to pheromones. Important theoretical discussions with simple applications are described in [5, 7], and [6] shows how these techniques can play a credible game of chess. The most elaborate model for synthetic pheromones is [17], which pursues the analogy to the point of constructing a synthetic chemistry through which agents interact with multi-typed pheromone traces.

The most mature practical use of pheromone techniques is in routing telecommunications packets (e.g., [2, 10]). Application of these techniques to moving physical entities can be traced to the Cascade system [13, 14], a self-routing modular material handling system. That work appealed to a neural backpropagation model rather than a pheromone model for its antecedents. However, the accumulation of link weights through backpropagation has many formal similarities to the

accumulation of pheromones on well-traveled paths. The application of these techniques to routing and load balancing is being extended in the ESPRIT MASCADA project [15]. Steels proposed similar mechanisms for coordinating small robots used in exploring remote planets [16]. Dorigo and colleagues [4, 8, 11] have applied these mechanisms to a range of optimization problems including the traveling salesperson problem and the quadratic assignment problem.

These various applications are NP-complete, but do not require the same level of inter-agent coordination that is seen in the Missionaries and Cannibals problem.

3. Algorithm and Metrics

This paper reports a synchronous stochastic algorithm in which agents evaluate their desire to move at fixed epochs widely spaced enough that the pheromone strength has time to stabilize. We demonstrate various combinations of three distinct pheromones: a bank pheromone that tells agents where they are, an undifferentiated population pheromone deposited by both Missionaries and Cannibals, and distinctive Missionary and Cannibal pheromones.

At each decision epoch, only those agents on the bank with the boat make a movement decision. Each such agent decides whether it wishes to move by evaluating a personal choice function *CF* that returns a real number in [0,1], evaluating a random variable uniformly distributed on [0,1], and comparing these two values. If the random number is less than the value of the choice function, the agent places itself on a list of candidates for movement. This random comparison provides the stochasticity analogous to the wandering behavior of insects.

The interesting details of the agent's decision are embedded in its choice function, which is a function of the levels of the available pheromones. In principle, each individual agent could have its own choice function, but in our experiments all Missionaries share one choice function and all Cannibals share another.

The number of agents that wish to move at any epoch may range from 0 to the total bank population. If no agents wish to move, none does, and the agents evaluate their choice functions and random number generators at the next epoch. If one or two agents wish to move, they do so. If more than two wish to move, two are chosen at random. This choice can be made without central control. For example, the agents race to the boat, and the first two that arrive are moved.

There is no guarantee that the boatload that results from this process will generate a safe state when the boat moves. We explore two approaches to handling this situation.

In Die Mode, we simply terminate the run at this point, recording a failure. A run that does not fail will eventually succeed when it reaches {R, 0, 0}. This mode suggests three performance metrics: the population of the right bank at the end of a run (whether successful or unsuccessful), the proportion of successes to total runs, and the length of successful runs.

In Continue Mode, the agents in the boat test for the safety of the trip before they leave the bank. If the transition is not safe, they disembark and wait for the next epoch. (Our Cannibals are aggressive enough to eat Missionaries if they have the chance, but responsible enough to follow the convention that requires them to leave the boat rather than transition to an unsafe state.) Under this mode, if a run terminates at all, it does so successfully, and the number of epochs required for success becomes the measure of effectiveness of the choice functions being considered.

Continue Mode implements a one-step explicit backtrack. In addition, our stochastic movement rules can return the entire system to a previous state, permitting more extensive backtracking under either option.

Given distinctive Missionary and Cannibal pheromones, agents who board the boat can detect an unsafe transition locally and without direct interagent communication. With more restricted pheromone vocabularies, a central observer must warn the voyagers so that they can leave the boat. This type of "watchdog" agent [12] detects global conditions invisible to local agents, but does not itself plan or execute actions in the domain. While watchdogs deviate from our ideal of a purely decentralized system, they entail far fewer of the disadvantages of centralization than do more intrusive centralized controls.

4. Experiments

We experiment with four levels of pheromone-based information.

4.1. Experimental Design

The major variable in our experiments is the pheromone vocabulary available to the agents.

1. As a baseline, we consider a world without pheromones. The choice function returns a constant probability and so generates a random walk (R).
2. Next, agents sense only a bank pheromone (B) that tells them the bank on which they are located.
3. Next, cannibals and missionaries lay down identical pheromones, so an agent can know its bank and the total population (BP), but not the individual number of missionaries and cannibals.

4. Finally, missionaries and cannibals lay down and sense distinct pheromones, in addition to the bank pheromone (BMC).

The design of a choice function in the second, third, and fourth cases has two steps. First, we must determine the form of the function, and then assign values to variables that produce the required probabilities. Part of our future program is using various optimization

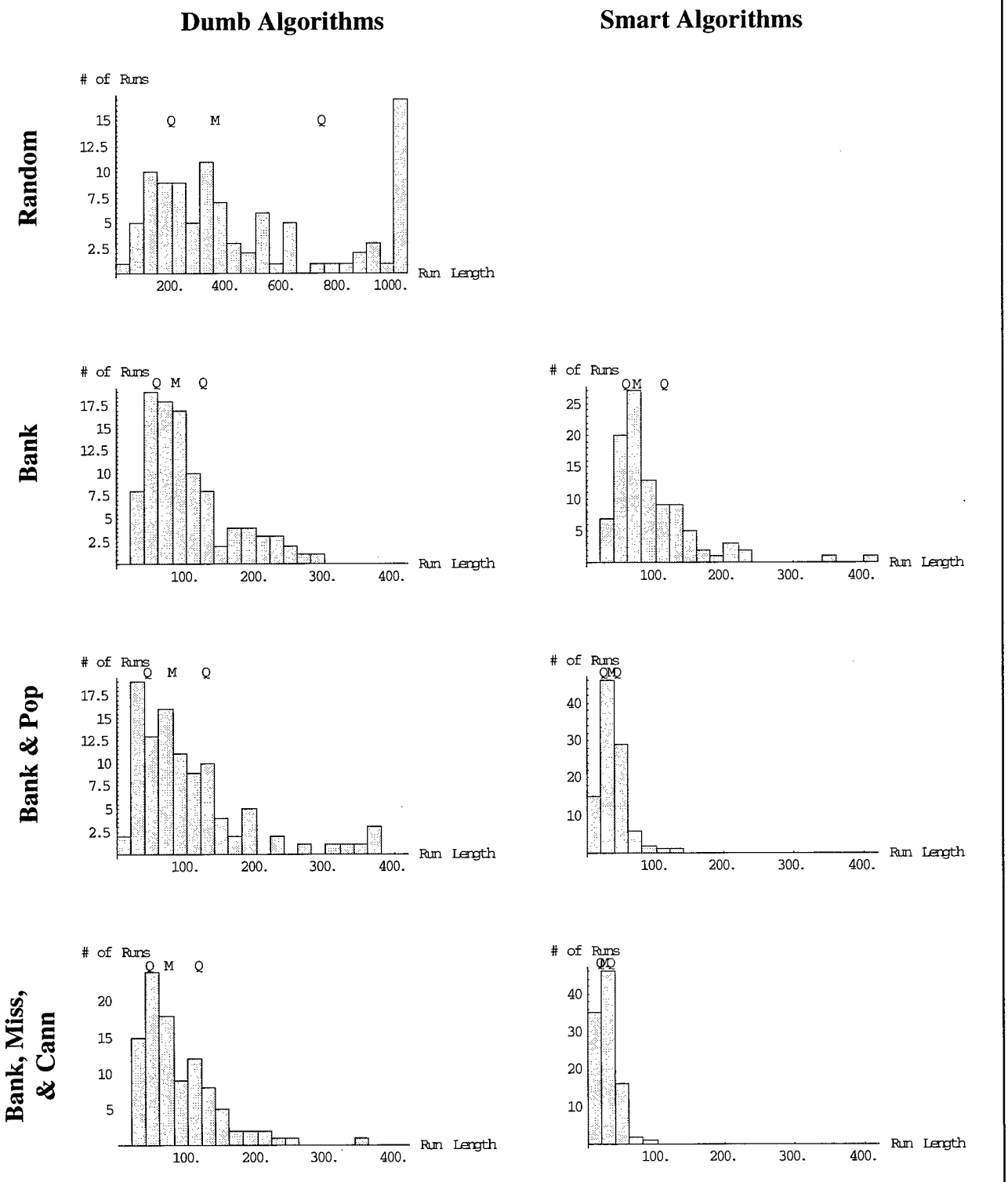


Figure 2: Summary of Continue Mode Run Lengths (100 Trials Each)

methods such as genetic processes or particle swarm optimization to tune these variables. In the work reported here, we set them manually. In each case we report two instantiations of the choice function. The first ("dumb") is intended as a lower bound, capturing only the basic rules of the game and the safety conditions for the state in which the agent thinks it is. The second ("smart") is intended as an upper bound, and embodies our knowledge of the global solution. We hypothesize that the result obtainable by automated tuning will fall between these extremes.

We use two of the metrics developed in the previous section: the number of successes per 10^5 trials in Die Mode, and the median run length over 100 trials in Continue Mode. We report medians and quartiles to give a feel for the data. Figure 2 summarizes for each configuration the distribution of Continue Mode run lengths. Each histogram records the M(edian) and upper and lower Q(uartiles) of its data. The highly skewed nature of these histograms emphasizes that we cannot use conventional parametric significance testing to assess the significance of differences between the various cases. We present a nonparametric analysis and discussion in a forthcoming paper, but here concentrate on exploratory conclusions suggested by the histograms.

4.2. Random Walk (R)

As a baseline, we define $CF = 0.5$, independent of bank or population. In Die Mode, this function does not succeed. In Continue Mode, the median number of epochs to solve the problem is 389.5, with quartiles of 191 and 655. The median number of times the system must back out of an unsafe transition is 162. The spike in the histogram in Figure 2 results from artificially terminating any runs longer than that value.

The performance may be sensitive to the transition probability. For $CF = 0.7$, the median time to solution is 849 steps, while for $CF = 0.3$ it is 602.5. However, the interquartile spreads range from over 250 for $CF = 0.5$ to over 600 for the last two cases, suggesting that the variation may be purely stochastic.

4.3. Bank Pheromone (B)

BDumb.—Adding a forcing function to attract agents from the left bank to the right might improve performance. We try a choice function that returns a higher probability for agents on the left bank than for those on the right. Sampling various pairs of probabilities shows a performance maximum in the region of *CF(left)* = 0.9, *CF(right)* = 0.3. Die Mode yields seven successes, with lengths ranging from 11 steps (the minimum) to 20.

	Left Bank			Right Bank		
	Pop.	**Miss.**	**Cann.**	**Pop.**	**Miss.**	**Cann.**
	2	0	2	**2**	0	1
	3	0	1	**3**	0	1
	4	2	0	**4**	1	1
	4	2	0	**4**	0	1
	5	0	2	**5**	0	1
	6	0	2	**6**	0	0
Total		4	7		1	5

Table 1: Population Dependencies in the Known Solution

In Continue Mode, the median time to complete is 85.5 steps, with quartiles at 57 and 124.

The Bank pheromone is necessary for reasonable performance of the algorithm. Configurations with other pheromones but without B perform essentially like R. While other pheromones can improve the performance of B, without it they can do nothing.

BSmart.—With a bank pheromone, we can encode some knowledge of the correct solution in the agents. Table 1 shows the populations that occur in the correct solution on each bank of the river, and the number of each kind of agent that gets in the boat in that state. Two different boat configurations occur for population = 4. If we define system states by bank and total population, missionaries move in fewer states than do cannibals. Thus we explore choice functions of the form *CF(Bank, AgentType)*.

Manual exploration of the parameter space suggests the function

CF(left, Missionary) = 0.7
CF(left, Cannibal) = 0.9
CF(right, Missionary) = 0.2
CF(right, Cannibal) = 0.4

This function produces only five successes in Die Mode, with two at 11 steps and the longest at 21. Continue Mode yields a median of 74 steps, with quartiles at 57 and 112. Figure 2 shows that the distribution of run lengths has shifted perceptibly to the left, though long outliers remain.

4.4. Bank and Population Pheromones (BP)

BPDumb.—Perhaps our agents can do better if we tell them the total population on their current bank. Our first set of choice functions for BP embeds only enough information to keep the agents safe. For example, if the population is six, at most one missionary should be allowed to move, since the boat can then only carry away a single cannibal to keep the numbers even, but any number of cannibals can move, yielding *CF(Left, Missionary, 6)* = 0.3 and *CF(Left, Cannibal, 6)* = 1. But

when the population is 1 or 2, everybody can move. The choice function for each bank and missionary is a vector of six probabilities. Where the safe condition is not clear, we revert to the probabilities *CF(Left)* = 0.9, *CF(Right)* = 0.3. Our overall vectors are:

CF(Left, Missionary) = <1, 1, .9, .9, .3, .3>
CF(Right, Missionary) = <1, .5, .3, .3, .3, 0>
CF(Left, Cannibal) = <1, 1, .9, .9, 1, 1>
CF(Right, Cannibal) = <1, .5, .3, .3, .5, 0>

In Continue Mode, population information adds very little to performance, with a median of 78 and quartiles of 44 and 128. However, this set of probabilities is much more robust in Die Mode, yielding 49 successes, including six with the optimal length of 11 and a maximum length of 21.

BPSmart.—If we permit ourselves knowledge of the correct solution, we can construct an even better choice function. For example, Table 1 shows that left-bank Missionaries should never move unless the population is 4, and left-bank Cannibals should never move at this population. Now our choice vectors are:

CF(Left, Missionary) = <1, 1, 0, 1, 0, 0>
CF(Right, Missionary) = <1, .5, 0, .5, 0, 0>
CF(Left, Cannibal) = <1, 1, 1, 0, 1, .7>
CF(Right, Cannibal) = <1, .5, .5, .5, .5, 0>

Figure 2 suggests that this function gives a dramatic improvement in performance over both BPDumb and BSmart. Continue Mode yields a median of 35.5 and quartiles of 23 and 44. Die Mode yields 5940 successes, with 194 at the minimum length and a maximum length of 59. The improvement over the dumb case is (78 – 35.5)/78 = 54%.

BPOther.—We have experimented with several choice functions for the BP case. It is instructive to consider our first candidate. In the correct solution, boats traveling from left to right always carry two passengers, while those traveling from right to left carry an average of 1.2 passengers. So we reasoned that right-bound probabilities should be *2/population*, while left-bound ones should be *1.2/population*. These functions yield no successful runs in Die Mode, with median lengths in Continue Mode on the order of 250, barely better than the random walk baseline and much worse than the bank-only case. This experience emphasis the challenge of developing pheromone-based algorithms for specific applications. Intuitions based on conventional approaches are often wrong, and there is no substitute for experimentation.

4.5. Bank, Missionary, and Cannibal Pheromones (BMC)

BMCDumb.—Finally, we distinguish missionary and cannibal pheromones. We represent the dependency on *mpop* and *cpop* in a table, as shown for missionaries in Table 2, where rows indicate the number of Cannibals (0-3) and columns the number of Missionaries (0-3). Each cell indicates the number of agents that could move from this bank to the other without creating an unsafe state. A similar table defines Cannibal choices. A Missionary's choice function (for example) returns the appropriate cell of the table divided by *mpop*. When all the Missionaries compare this value with independent uniform random [0,1] variates, on average the number of Missionaries that elect to move will be equal to the cell value.

If agents use the same table on both banks, the system will avoid unsafe states, but there will be no pressure toward moving all agents to the right bank. Bank pheromones enable agents to attenuate right-bank choice functions by a constant between 0 and 1 (in the results reported here, 0.7) to ensure that on average fewer agents move right-to-left than left-to-right.

This function actually under-performs both BP functions in Die Mode, yielding only 18 successes, with four at the minimal length of 11 and a maximum length of 20. As Figure 2 suggests, its Continue Mode performance is marginally better: the median is 73.5 with quartiles at 46 and 117. The ambiguity of the results suggests that we have not yet discovered the best way to use the distinct population levels.

BMCSmart.—Once again, we try embedding knowledge of the correct solution in the choice functions. Table 3 is an example of the table for a Missionary on the Left bank. The only non-zero cells correspond to states on one of the known trajectories from {L, 3, 3} to {R, 0, 0}, and the entry at such a cell is the number of Missionaries that should enter the boat to move to the next state in the trajectory. Similar tables define the behavior of right-bank missionaries, left-bank cannibals, and right-bank cannibals. Choice functions are computed as in BMCDumb, except that the tables explicitly record the difference between left and right bank, so no attenuation factor is needed. The cell-wise sum of the two tables from the same bank shows the total number of agents that should move. The set of four tables thus encodes the known solution to the problem. Consistent

		Missionary Population			
		0	**1**	**2**	**3**
Cannibal Population	**0**	0	0	0	0
	1	0	1	0	2
	2	0	0	2	1
	3	0	0	0	1

Table 2: Safe Choice Table for Missionary

with our insights from simpler choice functions, non-zero cells for the left bank are all 2, while those for the right bank are mostly 1, so that there is a net preference for left-to-right movement.

This formulation is promising, but a bit too simplistic. While the *expected* number of agents electing to cross at each epoch is according to the solution, the *actual* number at any given epoch may differ because of the stochastic nature of the search. As a result, the system can enter a safe state not on the correct trajectory, one for which both Missionaries and Cannibals have 0 entries in their tables, and the system has no way to backtrack. To overcome this problem, we set all entries in safe states that would otherwise be 0 to 0.1, generating a non-zero probability of movement from any safe state.

When we execute the system with this choice function, Die Mode succeeds 8939 times, 750 with the minimum length of 11, and a maximum length of 93. The Continue Mode median is 23.5, with quartiles at 18 and 34. The improvement over the dumb case with the same pheromones is (73.5 – 23.5)/73.5 = 68%, higher than in either B or BP.

	Die Mode			Continue Mode		
Case	# Successes	# at Len. 11	Max Len	Median	Lower Quartile	Upper Quartile
R	0	0	—	353	198	738
BDumb	7	1	20	85.5	57	124
BSmart	5	2	21	74	57	112
BPDumb	49	6	21	78	44	128
BPSmart	5940	194	59	35.5	23	44
BMCDumb	18	4	20	73.5	46	117
BMCSmart	8939	750	93	23.5	18	34

Table 4: Summary of Experimental Data

5. Results and Summary

Table 4 summarizes the results of our experiments. These results indicate:

- R is clearly inferior to all the others. Our algorithm is not just a random walk.
- The smart cases (those embedding knowledge of the correct solution) increase in performance as the pheromone vocabulary becomes riches. BPSmart outperforms BSmart, and BMCSmart outperforms them both. Figure 3 zooms in on the run-length histograms for the smart cases, clearly showing the improvement as the pheromone vocabulary increases. (Fifteen runs in the Bank-only case have been omitted by truncating the histogram at length 140.)
- The dumb cases show the same trend, but less clearly. BMCDumb is clearly superior to BDumb, but the differences between the other dumb cases are only suggestive.
- Smartness does matter. Figure 2 shows that with the possible exception of BSmart, smart algorithms outperform their dumb counterparts.

		Missionary Population			
		0	1	2	3
Cannibal Population	0	0	0	0	0
	1	0	1	0	2
	2	0	0	2	0
	3	0	0	0	0

Table 3: Ideal Choice Table for Left-Bank Missionary

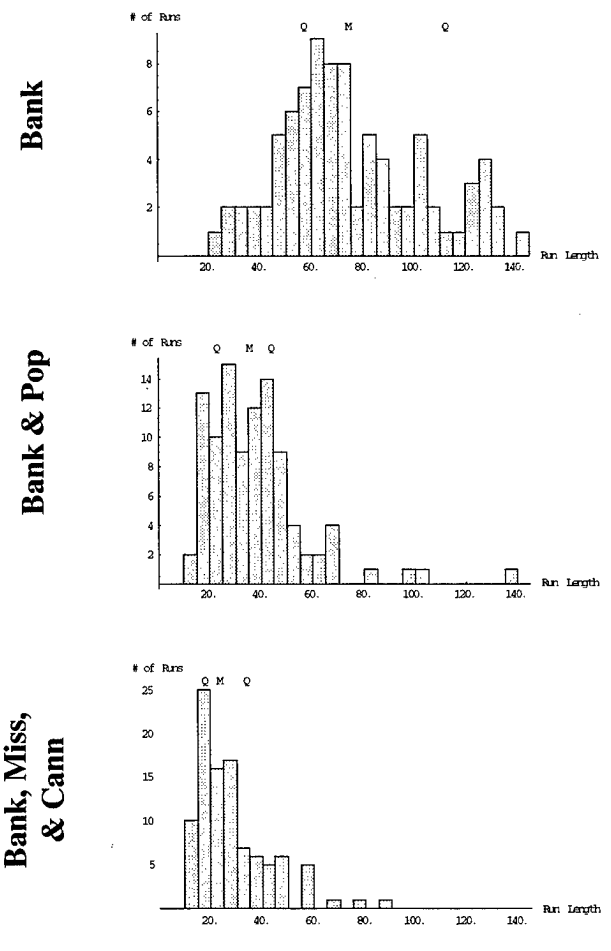


Figure 3: Continue Mode Run Lengths for Smart Algorithms

Figure 4 and Figure 5 show the trends for the number of successes in Die Mode. At $N=10^5$, the normal distribution is an excellent approximation to the binomial, so setting error bars in these figures at $\pm 1.96\sigma$, using the binomial value $\sigma = \mathrm{Sqrt}(pq/10^5)$, establishes a 95% confidence interval. Thus in these figures, overlap of the central point of one case with the error bars of another indicates that the two cases are not significantly different. These figures show that the Die Mode data yield many of the same conclusions as the Continue Mode data.

- R is significantly worse than all other cases.
- Smart cases increase in performance with pheromone vocabulary.
- The first two dumb cases show an increase in performance with more information, but the anomalous performance of BMCDumb emphasizes the difficulty of generating effective decision rules.
- Smart cases outperform corresponding dumb cases except for BSmart, which is indistinguishable from BDumb.

In summary,

1. Ants can solve the Missionaries and Cannibals problem without direct coordination among themselves, using only information that can be provided by a pheromone mechanism on each bank of the river and in the boat.
2. Pheromone-based search clearly outperforms random walk.
3. Decision mechanisms that encode known solutions outperform encodings that capture minimal knowledge about the domain.
4. The richness of the pheromone vocabulary improves the performance of this class of algorithms.

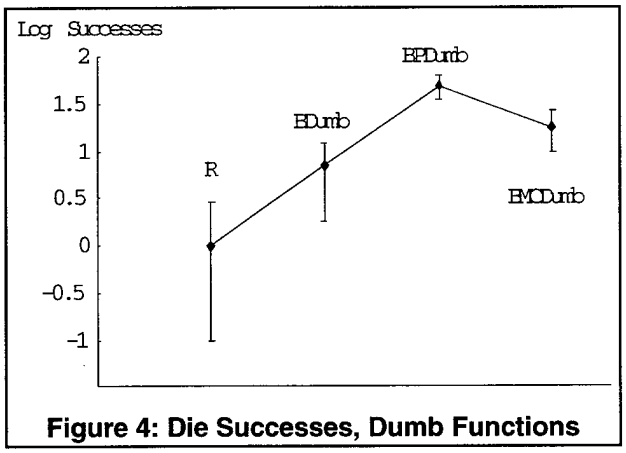


Figure 4: Die Successes, Dumb Functions

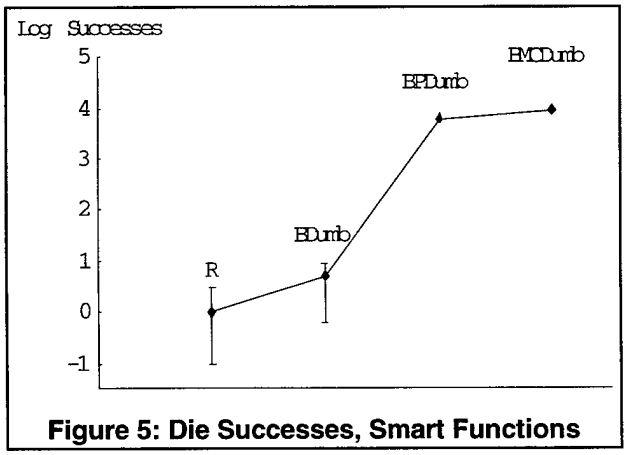


Figure 5: Die Successes, Smart Functions

The experiments reported here are initial results from more extensive programs underway at both ERIM CEC and DaimlerChrysler Research and Technology, in which we are exploring several more ambitious directions.

This algorithm is synchronous, permitting pheromones to stabilize between epochs. How does performance degrade if we do not allow as much time between epochs? We might stagger the decisions of individual agents, requiring that an individual agent wait long enough before moving for its own deposit to stabilize, but not necessarily wait for other agents to stabilize. Even if the number of epochs needed for successful runs rises, the actual elapsed time (in terms of pheromone deposit intervals) might decrease. Asynchroneity will require agents to react directly to pheromone levels, rather than first translating them to counts.

Genetic and similar methods can develop movement probabilities, using run lengths under Continue Mode as a useful fitness function. Intuitively, the probabilities derived from the known best traditional solution should give the best CF, but emergent mechanisms have a way of frustrating intuitions, and probabilities evolved directly against our mechanisms might outperform the manually derived ones. In addition, techniques for growing probabilities are critical for real problems whose "ideal" solutions are not known in advance.

In addition to the aggregation, evaporation, and diffusion functions found in physical environments, computational environments can support procedural and symbolic reasoning. We are extending the functions that places can apply to pheromones with such higher-level processes, known as "policies," that hybridize low-level agent-based pheromone mechanisms with high-level place-based processing, combining the efficiency and deployment benefits of lightweight agents with the enhanced performance potential of more sophisticated reasoning.

We are developing applications of these methods in two more realistic domains: the agile execution of air tasking orders (under the DARPA JFACC program) and material movement in agile manufacturing environments.

6. Acknowledgments

This work is supported in part by the DARPA JFACC program under contract F30602-99-C-0202 to ERIM CEC. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the US Government, or the DaimlerChrysler corporation.

7. References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. New York, Oxford University Press, 1999.

[2] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in Telecommunications Networks with "Smart" Ant-Like Agents. Santa Fe Institute, Santa Fe, NM, 1998.

[3] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Ph.D. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.

[4] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(1):1-13, 1996.

[5] A. Drogoul. *De la Simulation Multi-Agents à la Résolution Collective de Problèmes*. Ph.D. Thesis at University of Paris IV, 1993.

[6] A. Drogoul. When Ants Play Chess (Or Can Strategies Emerge from Tactical Behaviors? In *Proceedings of Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW '93)*, pages 13-27, Springer, 1995.

[7] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Harlow, UK, Addison Wesley Longman, 1999.

[8] L. M. Gambardella and M. Dorigo. HAS-SOP: Hybrid Ant System for the Sequential Ordering Problem. Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland, 1997. Available at ftp://ftp.idsia.ch/pub/luca/papers/SOP-tr-idsia-11-97.ps.gz.

[9] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized Shortcuts in the Argentine Ant. *Naturwissenschaften*, 76:579-581, 1989.

[10] M. Heusse, S. Guérin, D. Snyers, and P. Kuntz. Adaptive Agent-Driven Routing and Load Balancing in Communication Networks. *Advances in Complex Systems*, 1:234-257, 1998.

[11] V. Maniezzo, A. Colorni, and M. Dorigo. The Ant System Applied to the Quadratic Assignment Problem. IRIDIA, Université Libre de Bruxelles, Bruxelles, Belgium, 1994. Available at ftp://iridia.ulb.ac.be/pub/dorigo/tec.reps/TR.03-ANT-QAP.ps.gz.

[12] H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997. Available at http://www.erim.org/~van/gotoant.pdf.

[13] H. V. D. Parunak, J. Kindrick, and B. Irish. Material Handling: A Conservative Domain for Neural Connectivity and Propagation. In *Proceedings of Sixth National Conference on Artificial Intelligence*, pages 307-311, American Association for Artificial Intelligence, 1987.

[14] H. V. D. Parunak, J. Kindrick, and B. W. Irish. A Connectionist Model for Material Handling. *Robotics & Computer-Integrated Manufacturing*, 4(3/4):643-654, 1988.

[15] P. Peeters, P. Valckenaers, J. Syns, and S. Brueckner. Manufacturing Control Algorithm and Architecture. In *Proceedings of Second International Workshop on Intelligent Manufacturing Systems*, pages 877-888, K.U. Leuven, 1999.

[16] L. Steels. Cooperation between Distributed Agents through Self-Organization. Vrije Universiteit Brussel AI Laboratory, 1989.

[17] T. White and B. Pagurek. Towards Multi-Swarm Problem Solving in Networks. In *Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98)*, pages 333-340, IEEE Computer Society, 1998.

Constructing and Dynamically Maintaining Perspective-based Agent Models for Command and Control Applications in a Multi-Agent Environment

K. S. Barber J. Kim
The Laboratory for Intelligent Processes and Systems
Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
http://www.lips.utexas.edu
barber@mail.utexas.edu
phone: (512) 471-6152
fax: (512) 471-3652

Abstract

This paper describes a model that explicitly represents the declarative and behavioral knowledge of a goal-driven agent. The developed declarative and the behavioral models allow an agent to reason about itself, other agents, and the environment. The *declarative model* specifies data and services (capabilities) assigned to an agent while the *behavioral model* specifies the execution model of an agent (defined as states, transitions between states, and events affecting the transitions). An *Extended Statechart* (ESC) is used as the execution model. To maintain these models, a self-contained computational module called the *Perspective Modeler* is proposed and incorporated into the Sensible Agent Architecture discussed herein. With the Perspective Modeler, a Sensible Agent has the capability to model itself, other agents, and the environment to generate more desirable behaviors. In support of command and control applications, the Perspective Modeler maintains beliefs about very dynamic declarative and behavioral knowledge to 1) provide "what-is" assessments, 2) provide "what-if" assessments given projected events or actions, and 3) support deliberate actions or reactive actions. The Perspective Modeler implementation as a CORBA© object within the Sensible Agent Testbed allows distribution of the Perspective Modeler capability across multiple platforms. Additionally, the model representations (while populated with domain-specific information) and algorithms deployed in the Perspective Modeler are domain-independent promoting rapid and flexible integration into application systems.

1 Introduction

The blend of reactivity and deliberation is becoming widely accepted as a feasible approach to the representation of robust and flexible behaviors of agents within the multi-agent community [21;26]. Examples of implementations using this hybrid approach are PRS [14], TouringMachine [11], and INTERRAP [20]. However, achieving an optimal balance among an agent's reactive and deliberative behaviors is still an outstanding research issue [17]. For the detailed comparison of reactivity and deliberation, see [26].

In this paper, we propose a knowledge model to explicitly represent declarative and behavioral knowledge of an agent which will allow an agent to use this model to reason about itself, other agents, or the environment. The model (1) provides an agent with reactivity for response in dynamic and uncertain environments and (2) serves as an explicit symbolic model of the self-agent, other agents, and the environment for deliberative planning and conflict resolution.

These models are held in a self-contained computational module called the *Perspective Modeler* (PM). The PM is designed to support improved decision making in high level planners while providing potential reactive solutions to achieve robustness in the face of a dynamic environment, hardware failures, lack of communication bandwidth, or the time delays introduced

in end-to-end communication. The behavioral knowledge in the Perspective Modeler allows an intelligent agent to dynamically assess the current behavioral states and to model the possible future behavioral states of itself, other agents, and the environment. The declarative knowledge provides the agent with its best estimate of the goals, plans, beliefs, and resources of itself and others. The Perspective Modeler has been initially developed in the context of Sensible Agent Architecture [2], which is an on-going project in Laboratory of Intelligent Processes and Systems (LIPS) at the University of Texas at Austin.

The Perspective Modeler has been implemented in LOOM [7] and LISP as a CORBA© [22] object on top of the Inter-Language Unification system (ILU) [27]. The public interface for the module is defined in an IDL (Interface Definition Language), thus allowing other researchers easy access to the module. To investigate and test the functionality of the PM, it has been incorporated into the Sensible Agent Testbed [6] and applied to a naval frequency management problem. The Sensible Agent System functionality, including the Perspective Modeler, has been successfully demonstrated in many application domains [3;5;6].

In the next section, the Sensible Agent Architecture is briefly described. In Section 3, the PM is specified, followed by implementation of the PM in the naval frequency management domain, in Section 4. Section 5 draws the conclusions and addresses future work.

2 Sensible Agent

A Sensible Agent is a goal-driven entity capable of (1) deliberative or reactive planning and execution of one or more domain specific services or tasks, (2) maintaining and interpreting knowledge about states, events, and goals related to itself, other agents, and the environment, and (3) adapting its behavior according to the understanding of its own local goals and system goals [2]. A critical consideration for a Sensible Agent's behavior is the agent's level of autonomy (i.e. the types of organizational roles that exist in an agent's interactions with others).

The logical architecture of a Sensible Agent is composed of individual modules that address the functional requirements for an agent. Sensible Agents are composed of the following modules as shown in Figure 1.

- The Action Planner module is a general problem solver (planner) and executor in a Sensible Agent. This module interacts with the environment and other agents in the system. Domain-specific problem solving information, strategies, and heuristics are placed inside this module. The Action Planner is able to interpret, plan to achieve, and execute solutions for domain-specific goals. This module draws information from all other modules within the agent.
- The Autonomy Reasoner determines the appropriate autonomy level for each goal, assigns an autonomy level to each goal, and reports autonomy level constraints to other modules within the agent. The autonomy reasoner assigns one of three discrete autonomy levels: Command-driven, Consensus, and Master/Locally autonomous to specify how the agent will interact others to plan and achieve for goals.
- The Conflict Resolution Advisor identifies, classifies, and generates possible solutions for resolving conflicts occurring within the agent and between other agents. This module monitors the Action Planner and the Perspective Modeler to identify conflicts. It offers conflict resolution strategies to the Action Planner or the Autonomy Reasoner so that those modules may resolve the conflict.
- The Perspective Modeler maintains the knowledge about the self-agent, other agents, and the environment. A system designer can specify the desired services, data, and interfaces of each agent through this module. This module interprets internal or external events acting on the agent and changes its state accordingly. It also maintains beliefs about states and events external to the self-agent and predicts the actions of other agents and the environment. Other modules within the agent can access this module for necessary information (e.g. current or potential future states) and for changes that affect their reasoning processes.

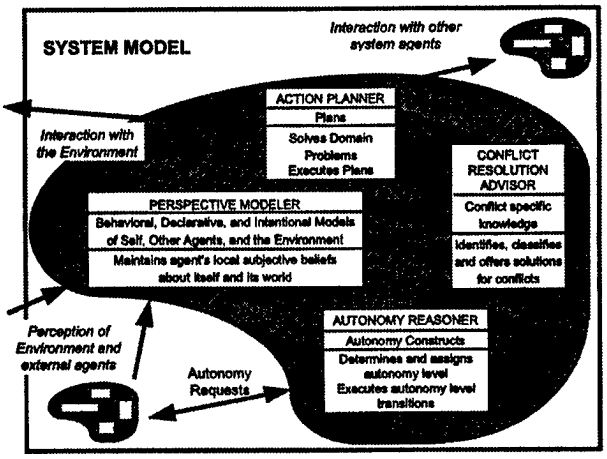


Figure 1. The Sensible Agent Architecture

3 Perspective Modeler

3.1 Assumptions

The Sensible Agent Architecture makes the following relevant basic assumptions:

- *Dynamic environment*: The environment may change over time independent of the actions of the agent. We cannot assume the environmental change will occur much slower than the speed of the reasoning mechanisms of an agent. Some level of reactivity may be necessary for an agent working in this environment.
- *Heterogeneous system*: There may exist more than one type of agent in the system. While all Sensible Agents in a system share system goals, heterogeneity among agents results when agents are designed such that they are responsible for solving different local goals unique to themselves and maintaining different behavioral models.
- *Local and subjective perspective*: While maintaining a consistent global view for each agent may be desirable, it has been pointed out [13;16;18] that the high communication and computational requirements in complex and dynamic environments makes this impractical. Thus the PM operates on the local perspective of an agent. It is important to note that this local perspective may not be consistent among all the constituent agents in the system. Additionally, the local perspective is a more manageable and flexible model for an agent to plan with. Global consistency will only be forced for the shared knowledge among agents.

3.2 Services

The Perspective Modeler (PM) provides the following services to other modules within a Sensible Agent:

- *Maintaining declarative and behavioral knowledge*: The PM functions as a database management system. When the PM perceives or is notified of changes in the characteristics of the self-agent, other agents, or the environment, it updates its internal declarative and behavioral knowledge accordingly to keep it current and locally consistent within the agent.
- *Providing what-is assessment*: Upon query from other modules (e.g. the Action Planner), the PM performs the function of a data server. The PM returns the best estimate of the declarative or behavioral knowledge about the self-agent, other agents, or the environment, to requesting agent modules.
- *Providing what-if assessment*: In this service, the PM performs the function of simulation by generating expected status (e.g. states) resulting from actions/events hypothesized by the other modules. The other modules can be informed of the potential future status of the self-agent, other agents, or the environment.
- *Supporting reactivity for given situation*: The agent's behavioral knowledge about itself defines how the agent responds to the given situation. When an environment in which a Sensible Agent is situated is very dynamic and the agent has very limited resources (e.g. computational resources or time), deliberate planning may become impractical. This functionality frees the planner (the Action Planner) from dealing with behaviors that can be handled by the reactive component of the PM.
- *Supporting dynamic behavioral representation*: Although an agent has a detailed explicit state space representation, it is not exhaustive. When an agent encounters a situation that is not fully specified, the requesting agent modules should generate appropriate behaviors using available declarative and behavioral knowledge. If the resulting behaviors are found useful, they will be incorporated into the existing behavioral knowledge for future use.

3.3 Architecture

There are two emerging approaches to drive the specification of reliable and robust agent systems [19]: (1) logic and (2) modeling techniques from software and knowledge engineering. We utilize modeling techniques since we believe it is critical to have a methodology that is flexible and sufficiently rich to allow designers to capture and represent domain-specific knowledge in the forms they observe in a real world environment.

The PM contains the agent's representation of itself, other agents, and the environment with which the agent interacts. This representation is specified in terms of the *declarative* and the *behavioral models*. The declarative model specifies (1) data and services (capabilities) assigned to an agent and (2) the attributes which characterize an agent. The behavioral model specifies the execution model of an agent (defined as the states, the transitions between those states, and the events affecting the transitions) that defines how to perform its designated tasks.

The four key components in the PM are the *Behavioral Modeler*, the *Declarative Modeler*, the *Event Monitor*, and the *Observation Correlator* as shown in Figure 2.

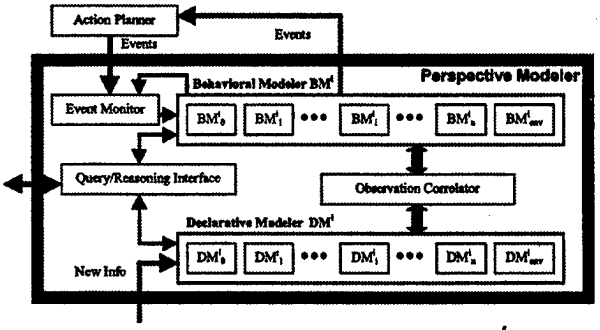


Figure 2. The Perspective Modeler PM^i of a Sensible Agent A^i

The Behavioral Modeler is composed of (1) a set of behavioral models describing the self-agent, other agents, and the environment and (2) the rules to maintain them. Similarly, the Declarative Modeler is composed of declarative models and maintenance rules. The Event Monitor looks for the events generated from the Action Planner (or the other agent modules that can generate events) and determines if those events correlate to the transitions in the behavioral models or the changes in the attribute values in the declarative models. Sensor perceptions or Action Planner actions may result in changes to the declarative model values. The Observation Correlator monitors the values of attributes in the declarative models to determine if these specific values or combinations of values correlate to the transitions in the behavioral models. It also notifies the Behavioral Modeler when it finds the current states in one of the behavioral models are invalid due to incorrect information.

A formal examination of the domain-specific services or tasks provides the necessary information to populate the representation constructs of the Behavioral Modeler and the Declarative Modeler. This examination is conducted using object oriented analysis and design techniques promoting modular agent-based system architecture. The formal engineering process consisting of knowledge acquisition, domain modeling, reference architecture (the object-oriented decomposition specifying what data and services an agent is responsible for), is applied [1]. The result of this effort is mapped to the Perspective Modeler's respective declarative and behavioral models.

3.4 Declarative Modeler

The Declarative Modeler contains a set of declarative models and the rules to maintain these models. A declarative model represents declarative knowledge, which is the knowledge about the data (in the forms of beliefs, goals, plans, and resources) and the services (capabilities) assigned to Sensible Agents.

A Sensible Agent A^i's Declarative Modeler DM^i is defined as follows:

$$DM^i =< DM^i_0, DM^i_1, ..., DM^i_j, ..., DM^i_n, DM^i_{env} >$$

where n is the number of agents being modeled and DM^i_j represents A^i's local perspective of the declarative model of A^j. DM^i_i is the declarative model of the self-agent and DM^i_{env} is A^i's local perspective of the environment. We assume that the environment does not have any specific goals or plans, and the declarative model of the environment has only resources.

Figure 3 shows the specification of elements of a declarative model in the Declarative Modeler. A declarative model contains beliefs, goals, plans, and resources.

Possible sources for new information (beliefs, goals, plans, and resources) are (1) observation, (2) communication with other agents, and (3) rules associate with beliefs. To maintain the declarative models, we implement a simple knowledge update algorithm, which has two rules.

1. If a fact B is perceived by the self-agent or asserted by the other agent in the same group (or team), the fact B is believed.
2. If no other agent tells the self-agent its belief B is invalid and the self-agent cannot perceive a counter example to the belief B, the self-agent continuously believes B.

Belief	::= atom
Goal	::= <GoalID, GoalType, ListOfOwner, PreCond, PostCond, ListOfConstraint, ListOfResource>
Plan	::= <PlanID, ListOfAction, PreCond, PostCond, ListOfConstraint, ListOfResource, Cost>
ListOfAction	::= Action|<ListOfAction>
Action	::= <ActionID,PreCond, PostCond,ListOfResource>
Resource	::= <ResourceID, ResourceType, ListOfOwner, CurrentValue, ListOfConstraint>
ResourcesType	::= <Transferable|NonTransferable> <Depletable|NonDepletable>

Figure 3: Specification of elements in a declarative model

3.5 Behavioral Modeler

The Behavioral Modeler contains a set of behavioral models and the rules to maintain these models. A behavioral model represents the behavioral knowledge, which is the knowledge about the execution model of an agent (defined as the states, the transitions between these states, and the events affecting the transitions).

There have been a series of efforts to represent behavioral knowledge of distributed systems using object-oriented technique [9;10;12;23;24]. Among them, state charts are considered to be a feasible approach for behavior modeling [17].

We employ Extended Statecharts (ESCs) [25], a derivative of Harel's Statecharts [15] with respect to temporal and hierarchical extensions, as well as soft failure models, as a comprehensive modeling mechanism for the behavioral models of an agent. ESCs allow for the explicit representation of declarable problem-specific agent *soft failures*, thereby allowing failure-related information to be incorporated within the high-level system design. ESCs exploit the XOR state representation in Statecharts to integrate failure information in a high-level system representation. Combined with a powerful event and transition structure this modeling technique allows the development of detailed high level specification of agent behaviors. An ESC is a 4-tuple (S, E, G, T), where, S is a set of states, E is a set of events, G is a set of guards and T is a set of transitions. The set of states S is defined as

$$S = \bigcup_{i=1}^{n} S^i$$

where, n is the number of levels in the model hierarchy. Exit-safe states represent states wherein a sub-state is in a stable state to process the transition out of a parent state. That is, events causing a transition between higher level (parent) states are handled only if the sub-state of the parents' state is an exit-safe sub-state. This extension explicitly allows the developer to specify certain non-interruptible critical operations. Non exit-safe states are states that are not exit-safe and hence do not allow for higher-level state transitions.

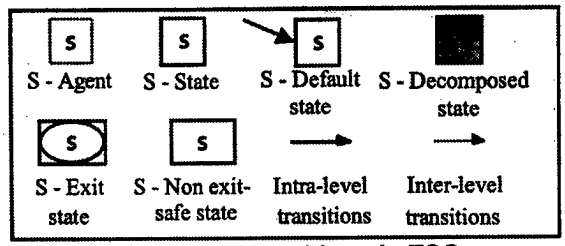


Figure 4. States and transitions in ESC graphical representation

Figure 4 illustrates the graphical ESC notations. The dashed rectangle represents the definition of an agent that is a domain-specific definition. The solid rectangle is used to represent a state that is decomposed immediately. The arrow pointing to a state denotes a default state that the agent reaches first among other states at the same level. The shaded rectangle is used to represent decomposed states. This means that the state S is decomposed further in another diagram. Two types of transitions are distinguished to manage the hierarchical information transfer between states. These include: (1) Intra-level transitions (T_{intra}): transitions between states in the same hierarchical level and (2) Inter-level Transitions (T_{inter}): transitions between states in different hierarchical levels. Intra-level transitions are used to represent normal behavior and inter-level transitions are used to represent abnormal behavior.

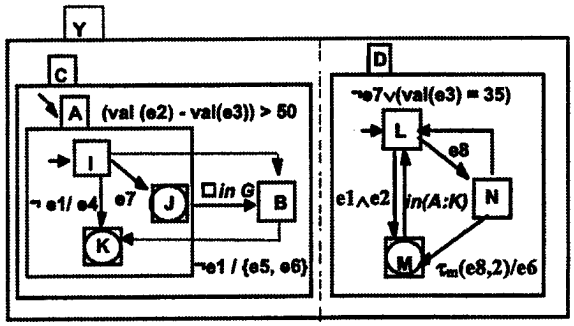


Figure 5. ESC example graphical representation

Figure 5 shows an example ESC representation. In the figure, **Y** is an *agent* that has states **C** and **D** in an AND composition. State **C** has sub-states **A** and **B** that are in a XOR combination, **A** being the default state. **D** has sub-states **L**, **M** and **N** where **L** is the default state. Notice that **J**, **K** and **M** are exit-safe states. Therefore, a transition from **Y** to another higher level parent state may occur if and only if the sub-state of **C** is either **J** or **K** and if sub-state of **D** is **M**.

A Sensible Agent A^i's Behavioral Modeler BM^i is defined as follows:

$$BM^i = < BM^i_0, BM^i_1, ..., BM^i_i, ..., BM^i_n, BM^i_{env} >$$

where n is the number of agents being modeled and BM^i_j represents A^i's local perspective of the behavioral model of A^j. BM^i_i is the behavioral model of the self-agent and BM^i_{env} is A^i's local perspective of the environment. While an agent A_i may possess a very accurate and detailed behavioral model BM^i_i of itself, it is difficult to expect the same level of accuracy for any BM^i_j ($i \neq j$), which is a behavioral model of another agent or the environment. The behavioral models are defined *a priori*

by the designer, and can be changed in the course of system operation through observation or speech acts.

State	::= <StateID, StateType, PreCond, PostCond, ListOfTransitions, BeliefFocus, ListOfDefaultAction, SuperState, SubState>
StateType	::= ExitSafe|NonExitSafe|Default|Decomposed
BeliefFocus	::= ListOfBelief
ListOfDefaultAction	::= ListOfAction
Transition	::= <TransitionID, TransitionType, FromState, ToState, PreCond, PostCond>
TransitionType	::= InterLevel|IntraLevel

Figure 6. Specification of the elements in a behavioral model

The specification of elements of a behavioral model in the Behavioral Modeler is shown in Figure 6. This specification follows the definition of ESC and the *belief focus* is attached to each state.

Definition: A *belief focus* is a set of beliefs that must hold true for a given state. A state *S* is the current state if and only if its *belief focus* is valid.

According to ESC's inheritance mechanism, the *belief focus* for a parent-state is inherited by its sub-states. The Observation Correlator uses the *belief focus* to maintain the consistency between declarative and behavioral models.

3.6 Correlating the DM and BM

Figure 7 shows the algorithm that handles state transitions in each behavioral model in the Behavioral Modeler. The Observation Correlator uses this algorithm to update every behavioral model it is maintaining.

However, since we only have local perspective, the state that an agent believed to be the current state might not actually be the current state. All subsequent reasoning processes under this faulty assumption may be doomed to fail. To offset this, the Observation Correlator utilizes the belief focus of the current state. The belief focus must be always valid. The Observation Correlator continuously monitors the *belief focus* of the current state of each behavioral model. If it finds the belief focus becomes invalid, it initiates consistency check.

```
for an event e,
  foreach behavioral model, b
    foreach current state s,
      foreach transition t that is an outgoing transition from s
        if e satisfies the precondition of t
          if all of the current sub-states are exit-safe OR e is a special event type that exits a state unconditionally
            then
              take transition t
            else
              return
            endif
          else
            pend the event to the current state
          endif
        endforeach
      endforeach
  endforeach
end
```

Figure 7. Algorithm for state transitions

```
begin
  while the current state, s, is not the root state
    s <= the parent state of s
    if the belief focus of s is valid
      then breakwhile
    endif
  endwhile
  if s is the root state
    then fail
  endif
  while there exist the sub-states of s
    foreach sub-states of s, c
      if the belief focus of c is valid
        then
          s <= c
          breakforeach
      endif
    endforeach
  endwhile
end
```

Figure 8. Algorithm to find the current state of a behavioral model

Since sub-states inherit the *belief focus* of the parent state, we can search the current state by climbing the ESC hierarchy very efficiently as shown in Figure 8.

The violation may result from incorrect observations, faulty communications, or inaccurate behavioral models, etc.

4 Example Domain

An implementation of the Perspective Modeler has been applied to the naval radar frequency management domain. The following subsections describe this application domain and the simulation.

4.1 Problem Description

Naval radar frequency management domain consists of a set of geographically distributed ships, each with one radar. The ships must maintain position and frequency relationships such that radar interference is minimized. Radar interference is any form of signal energy detected by a radar that comes from some source other than a reflection of its own emitted wave and obscures actual return signals. A radar that detects interference is called a victim radar. Radar interference decreases the signal to noise ratio of a victim radar, thereby making it more difficult for this radar to detect targets.

Radar interference occurs primarily when two radars are operating in close proximity at similar frequencies. A radar that is causing interference is called the source radar. Sources of interference for a victim radar in the naval domain include other friendly naval radars as well as radars of enemy and neutral entities. A source radar can cause interference for a victim without detecting any interference to its own operation. We explore two possible methods of eliminating radar interference, changing a radar's frequency or location. Unfortunately, the positions of each ship have been determined by tactical considerations and are not an available parameter for modification.

4.2 Implementation

The behavioral and declarative model of a radar agent were elicited through a formal specification using domain experts [4].

Figure 9 shows the resulting behavioral model of a Radar agent. It has two main XOR states, which are *Normal* state and *Error* state. *Normal* is the default state of the Radar agent. *Radar Off* is the default state of *Normal*. *Stand by* is the default state of *Radar Off*. When the radar receives the NewTarget event (e1), it transitions to the *Slewing* state from the *Stand by* state in *Radar Off*. In the *Slewing* state, following the default actions attached to this state, the radar motors engage and move the radar

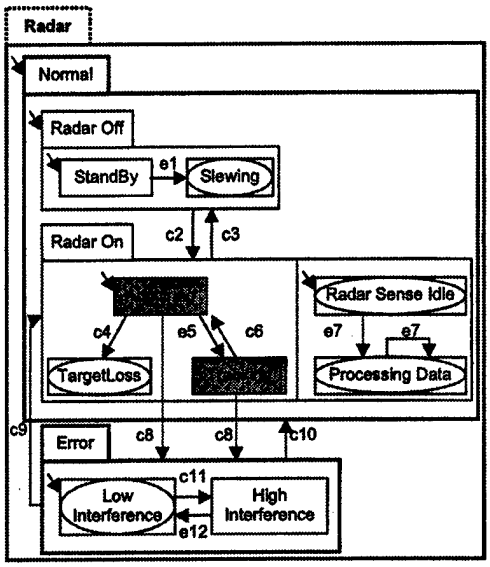


Figure 9. The behavioral model of Radar agent

to the general location of the target. The event, RSlewDone (c2), signifies that the radar is in the correct position. Upon this event, the radar enters the *Radar On* state. In the *Radar On* state, the radar simultaneously enters the concurrent default states of *Acquiring* and *Radar Sense Idle*. When the radar has the range on the target, signified by the HaveRange event (e5), the radar transitions to the *Tracking* state. If the radar loses the range on the target, signified by the LostRange event (c6), the radar returns to the *Acquiring* state. If the radar has lost the target completely, signified by the composite event c4, it transitions to the *Target Lost* exit-safe state and produces the event c3. Event c3 then allows the transition from *Target Lost* to the *Stand By* state of the *Radar Off* state. The radar agent can fail when it gets interference from other emitters/radars. The radar can function normally if the interference is low (soft failure) and fails to perform well if the interference is high (hard failure). Whenever the radar detects any interference (c8), in the *Acquiring* and the *Tracking* states, it transitions to the *Error* state and defaults to the *Low Interference* state. These transitions are caused by failures and therefore represented as inter-level transitions. Event ControlInterf (c11) transitions the *Low Interference* state to the *High Interference* state. In this state, the radar tries to control the interference by reducing it. After the radar sets the interference to a low value (e12), the agent transitions from the *High Interference* state to the *Low Interference* state. If the radar can perform normally with an interference less than a variable t_2 (c9), the agent transitions to the *Acquiring* state of the *Radar On* state.

Figure 10. Simulation of three Sensible Agents working in a dynamic enviornment.

Figure 10 shows the simulation of three Sensible Agents on a mission in the Persian Gulf. The weather conditions may prevent them from communicating with each other and from coordinating their actions in order to avoid radar interference. The charts in the lower left of the screen show the interference levels of each agent and the frequencies they are operating at. The scenario in this example begins with a group of three ships proceeding to the point. For each Sensible Agent, plans for radar frequency assignments are developed, actions are executed, conflicts are detected and resolved, and autonomy levels are assigned based on the information in the Perspective Modeler of each Sensible Agent.

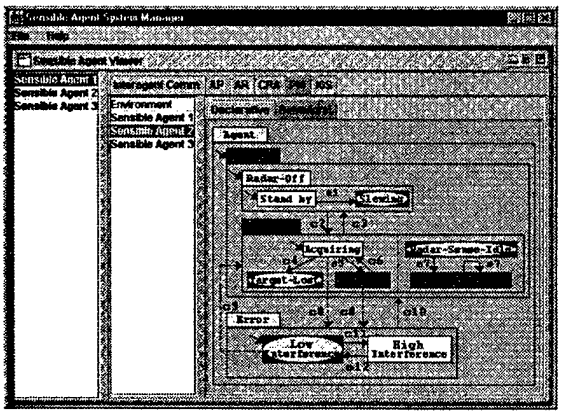


Figure 11. Behavior model BM^1_2 of Sensible Agent A^1

Figure 11 shows the behavioral model BM^1_2 in BM^1, meaning that the Sensible Agent A^1's local perspective of the behavioral model of Sensible Agent A^2. A^1 believes that the current state of A^2 is *Tracking* and *Processing Data* state.

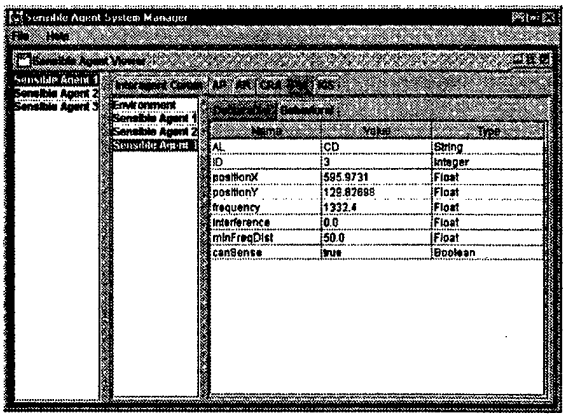


Figure 12. Declarative model DM^1_3 of Sensible Agent A^1

Figure 12 shows the declarative model DM^1_3 in DM^1. This represents the Sensible Agent A^1's local perspective of the declarative model of Sensible Agent A^3. DM^1_3 shows the attributes of A^3 known to A^1 so far. It often is incomplete and might not actually be correct.

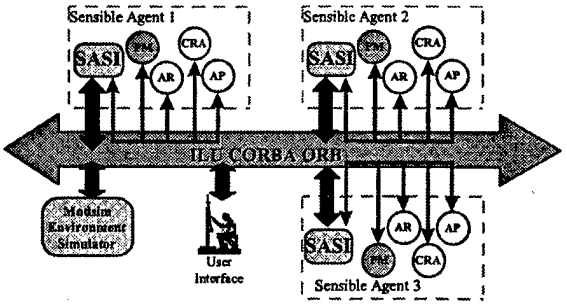


Figure 13. Implementation Architecture of Sensible Agents

Figure 13 shows the implementation architecture of Sensible Agent System for the given problem domain including PM. The SASI (Sensible Agent System Interface) is a add-on module for interfacing a Sensible Agent to the simulation environment. It interprets the information from the simulator and imports it to each agent's Perspective Modeler. ModSim [8] is used as a simulation engine.

PM is implemented in LOOM [7] and LISP running on Solaris machine. A Graphical User Interface (GUI) written in Java has also been developed to show the functionality of the PM. The GUI also allows the agent designer to change the DM online so that he or she can minimize the cost/time in developing the system. The Java GUI module and other Sensible Agent modules can connect to the PM module through the Inter-Language Unification system (ILU) [27] which is OMG CORBA©

compliant ORB. All modules are implemented as CORBRA© objects, so that they can be running on different platforms and different machines.

5 Conclusion

In this paper, we proposed explicit knowledge models to represent declarative and behavioral knowledge of a goal-driven agent. An agent can use this model to reason about itself, other agents, or the environment. The representation is specified in terms of the declarative and the behavioral models. The declarative model specifies data and services (capabilities) assigned to an agent while the behavioral model specifies the execution model of an agent (defined as the states, the transitions between those states, and the events affecting the transitions). The Extended Statechart has been employed as the execution model. A self-contained computational module called the Perspective Modeler maintains these models and is incorporated into the Sensible Agent Architecture. With this module, a Sensible Agent is capable of modeling itself, other agents (not limited to Sensible Agents), and the environment to generate more desirable behaviors. For the naval frequency management problem, the PM was shown to model and maintain the declarative and behavioral execution of radars (agents) detailing the agent's perspective of itself, other radars (agents), and the environment. The entire system was implemented and functionally tested successfully. Future research is required to seek to quantify the performance of the PM in this domain.

6 Reference

[1] Barber, K. S. 1996. The Architecture for Sensible Agents. In *Proceedings of the International Multidisciplinary Conference, Intelligent Systems: A Semiotic Perspective*, (Gaithersburg, MD: National Institute of Standards and Technology), 49-54.

[2] Barber, K. S., Goel, A., Graser, T. J., Liu, T. H., Macfadzean, R. H., Martin, C. E., and Ramaswamy, S. 1997. Sensible Agents. In *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, (Orlando, FL: .

[3] Barber, K. S., Goel, A., Han, D., Kim, J., Liu, T. H., Martin, C. E., and McKay, R. M. 1999, Problem-Solving Frameworks for Sensible Agents in an Electronic Market. In *Multiple Approaches to Intelligent Systems: Proceedings of the 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-99, Cairo, Egypt, May 31 - June 3, 1999*, Lecture Notes in Artificial Intelligence: Subseries of Lecture Notes in Computer Science, Imam, I., Kodratoff, Y., El-Dessouki, A., and Ali, M., Eds. (Berlin: Springer), 470-479.

[4] Barber, K. S., Graser, T. J., Jernigan, S. R., McGiverin, B. J., and Roesler, A. 1998. The Systems Engineering Process Activities: The Methodology and Supporting Tools, Technical Report, TR98-UT-LIPS-SEPA-01, The University of Texas at Austin, Austin, TX.

[5] Barber, K. S., McKay, R. M., Martin, C. E., Liu, T. H., Kim, J., Han, D., and Goel, A. 1999, Sensible Agents in Supply Chain Management: An Example Highlighting Procurement and Production Decisions, in *19th ASME Computers and Information in Engineering Conference, Internet-Aided Design, Manufacturing, and Commerce Technical Comittee*. Las Vegas, NV, pp. To be published.

[6] Barber, K. S., White, E., Goel, A., Han, D., Kim, J., Li, H., Liu, T. H., Martin, C. E., and McKay, R. 1998. Sensible Agents Problem-Solving Simulation for Manufacturing Environments. In *Proceedings of the AAAI SIGMAN Workshop on Artificial Intelligence and Manufacturing: State of the and State of Practice*, (Albuquerque, NM: AAAI Press), 1-8.

[7] Brill, D. 1993. LOOM Reference Manual, , Ver 2.0, University of Southern California, Los Angeles, CA.

[8] Caci. 1997, MODSIM III, , vol. 1999. http://www.modsim.com/: Caci Products Co.

[9] Cherry, G. W. 1993. Stimulus-Response Machines: A New Visual Formalism for Describing Classes and Objects. *ACM SIGSOFT Software Engineering Notes 18*(2): pp. 86-95.

[10] Clements, P. C., Heitmeyer, C. L., Labaw, B. G., and Mok, A. K. 1993. Applying Formal Method to and Embedded Real-Time Avionic System. In *Proceedings of the First IEEE Workshop on Real-Time Applications*, (New York: pp. 46-49.

[11] Ferguson, I. A. 1992, Toward an Architecture for Adaptive, Rational, Mobile Agents. In *Decentralized A.I. 3 : Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Kaiserslautern, Germany, August 5-7, 1991*, Werner, E. and Demazeau, Y., Eds. (Amsterdam: Elsevier Science Publishers), 249-261.

[12] Gabrielian, A. and Franklin, M. K. 1991. Multilevel Specification of Real Time Systems. *Communications of the ACM 34*(5): pp. 51-60.

[13] Gasser, L. 1992, Boundaries, Identity, and Aggregation: Plurity Issues in Multiagent Systems. In *Decentralized A.I. 3 : Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Kaiserslautern, Germany, August 5-7, 1991*, Werner, E. and Demazeau, Y., Eds. (Amsterdam: Elsevier Science), 199-213.

[14] Georgeff, M. P. and Ingrand, F. F. 1989. Decision-making in an embedded reasoning system. In *Proceedings of the Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, (Detroit, MI.: 972-978.

[15] Harel, D. 1987. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming 8*: 231-274.

[16] Jennings, N. R. 1992, On Being Responsible. In *Decentralized A.I. 3 : Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Kaiserslautern, Germany, August 5-7, 1991*, Werner, E. and Demazeau, Y., Eds. (Amsterdam: Elsevier Science), 93-102.

[17] Malec, J. 1995, A Unified Approach to Intelligent Agency. In *Intelligent Agents: ECAI-94 Workshop on Agents Theories, Architectures, and Languages*, Wooldridge, M. J. and Jennings, N. R., Eds. (Berlin: Springer-Verlag), 233-244.

[18] Mason, C. L. and Johnson, R. R. 1989, DATMS: a Framework for Distributed Assumption Based Reasoning. In

Distributed Artificial Intelligence II, Gasser, L. and Huhns, M. N., Eds. (London: Pitman Publishing), 263-317.

[19] Mulder, M., Treur, J., and Fisher, M. 1997. Agent Modelling in METATEM and DESIRE. In *Proceedings of the 4th International Workshop, ATAL'97*, (Providence, Rhode Island, USA: 193-207.

[20] Müller, J. P., Pischel, M., and Thiel, M. 1995, Modeling Reactive Behaviour in Vertically Layered Agents Architectures. In *Intelligent Agents: ECAI-94 Workshop on Agents Theories, Architectures, and Languages*, Wooldridge, M. J. and Jennings, N. R., Eds. (Berlin: Springer-Verlag), 261-276.

[21] O'Hare, G. M. P. and Jennings, N. R. 1996.Foundations of Distributed Artificial Intelligence, (New York).

[22] OMG. 1999, OMG CORBA Home Page, , vol. 1999: Object Management Group.

[23] Richter, G. and Maffeo, B. Toward a Rigorous Interpretation of ESML-Extended Systems Modeling Language. *IEEE Transactions on Software Engineering 19*(2): pp. 165-180.

[24] Rumbaugh, J. 1988. State Trees as Structured Finite State Machines for User Interfaces. *Communications of the ACM 10*(15): pp. 15-29.

[25] Suraj, A., Ramaswamy, S., and Barber, K. S. 1997. Extended State Charts for the Modeling and Specification of Manufacturing Control Software. *International Journal of Computer Integrated Manufacturing, Special Issue on Design and Implementation of Computer-Integrated Manufacturing Systems: Integration and Adaptability Issues 10*(1-4): 160-171.

[26] Wooldridge, M. J. and Jennings, N. R. 1995, Agent Theories, Architectures, and Languages: A Survey. In *Intelligent Agents: ECAI-94 Workshop on Agents Theories, Architectures, and Languages*, Wooldridge, M. J. and Jennings, N. R., Eds. (Berlin: Springer-Verlag), 1-39.

[27] Xerox. 1998, Inter-Language Unification -- ILU, , vol. 1999: Xerox/PARC.

Rapid Integration and Coordination of Heterogeneous, Distributed Agents for Collaborative Enterprises

David V. Pynadath, Milind Tambe, Nicolas Chauvat
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{pynadath,tambe,nico}@isi.edu

Abstract

As the agent methodology proves more and more useful in organizational enterprises, research/industrial groups are developing autonomous, heterogeneous agents that are distributed over a variety of platforms and environments. Rapid integration of such distributed, heterogeneous agent components could address large-scale problems of interest in these enterprises. Unfortunately, rapid and robust integration remains a difficult challenge. To address this challenge, we are developing a novel teamwork-based agent integration framework. In this framework, software developers specify an agent organization through a team-oriented program. To locate and recruit agent components for this organization, an agent resources manager (an analogue of a "human resources manager") searches for agents of interest to this organization and monitors their performance over time. TEAMCORE wrappers render the agent components in this organization team ready, thus ensuring robust, flexible teamwork among the members of the newly formed organization. This implemented framework promises to reduce the development effort in enterprise integration while providing robustness due to its teamwork-based foundations. We have applied this framework to a concrete, running example, using heterogeneous, distributed agents in a problem setting comparable to many collaborative enterprises.

1. Introduction

An increasing number of collaborative enterprises are turning to agent technology to address the complex, dynamic environments common to most enterprises [1, 4]. As more agents populate our organizations, whether in businesses, the military, etc., there is a more critical need to rapidly marshal the agents needed for newly arising tasks and to enforce proper coordination among these agents. Unfortunately, although the agent methodology does provide a great simplification over directly integrating legacy systems, the problem of coordinating agent behavior in a large-scale system remains difficult.

First, in the distributed, open environments found in most enterprises, there are great difficulties in identifying and accessing all of the agents relevant to the particular information and control needs of a new task. Second, since the recruited agents are not usually built to work together, building an integrated system with appropriate coordination among the agent components is difficult. Third, the resulting integrated system must be robust, in that it ensures the desired output despite the uncertainties of a dynamic, open environment. There are other issues in integration as well (e.g., common communication language), but this short note focuses on only the three key challenges mentioned here.

To address these integration challenges, our TEAMCORE project focuses on enabling enterprise managers to build large-scale agent organizations. There are currently two key aspects to this project. The first focuses on the creation, specification, and monitoring of the agent organization. The second focuses on enabling the organization to reliably execute tasks, by ensuring robust teamwork among the agents in the organization.

KARMA (*Knowledgeable Agent Resources Manager Assistant*) addresses the first aspect by assisting enterprise managers in three ways. First, Karma aids in *team-oriented programming*, where the system designer specifies a hierarchical agent organization, as well as its high-level goals (e.g., plan hierarchy, supply chain). Team-oriented programming abstracts away from coordination details, thus eliminating the burden of writing large numbers of coordination plans. Second, Karma locates agents that match the specified organization's requirements and assists in allocating organizational roles to such agents, thus alleviating the burden of searching through the vast numbers of agent components present in most enterprises. Third, Karma monitors the organization to diagnose failures and to evaluate agent performance for future (re)organizations. Karma's agent resources management functionality differs significantly from *middle agents* such as matchmakers. If middle agents are the analogues of the "middle-men" of physical commerce [2], then agent resource managers are the analogues of the "human resources managers" of a commercial firm. Agents such as Karma will become increasingly critical with the increasing number of agent components available within an enterprise (and eventually across multiple enterprises).

Once the enterprise manager has specified a team-oriented program, the second aspect of the TEAMCORE

project focuses on robust execution. Agent teamwork enhances robust execution, since we can expect agent components, as team members, to act responsibly towards one another, covering for each other's execution failures and sharing key information. To enable such teamwork among independently designed agent components, we make the agents *team-ready* by providing each with a separate TEAMCORE wrapper. We thus avoid the need to modify the agents themselves, an important consideration when the agent components can be complex legacy systems. The TEAMCORE wrappers are based on STEAM, a reusable, general-purpose teamwork module that encapsulates reasoning about common teamwork coordination, including contingencies in such coordination [6]. Given this model, the TEAMCORE wrappers automatically generate the required coordination actions in executing a team-oriented program. Thus, TEAMCORE shields the human developer from the responsibility of designing all such specifications.

We have applied our framework to a concrete problem from a military domain that shares many features with most enterprise integration tasks. In our example problem, we successfully integrated various information and control agents to provide a simulated mission rehearsal of the evacuation of civilians stranded in a hostile area. The integrated system had to enable a human commander to interactively provide locations of the stranded civilians, safe areas for evacuation, and other mission parameters, and to then have simulated helicopters fly a coordinated mission to evacuate the civilians. The integrated system plans routes to avoid known obstacles, obtains information about dynamic enemy threats, and changes routes when needed. Thus, we have an information and task dependency structure similar to the supply chains found in many collaborative enterprises.

Our framework enabled the construction of such an integrated system out of 11 different existing components, including a multi-modal user interface agent, a route-planner, a web-querying information-gathering agent and synthetic helicopter pilots. Different developers, using four different computer languages, designed these agents, which ran on two different operating systems, on machines distributed across the United States. The heterogeneity and decentralization present in this military enterprise reflect the reality of most collaborative enterprises. Here, we used Karma to specify the necessary team-oriented program and to successfully locate relevant agents through an interface with a matchmaker and other middle agents. The chosen agents, with their TEAMCORE wrappers, successfully executed the team-oriented program, with this execution being robust against agent failures and other dynamic events similar to those that arise in most real-world enterprises.

2. Karma & TEAMCORE

Figure 1 illustrates how one builds agent organizations through the Karma-TEAMCORE framework. The numbered arrows show the typical stages of this process. In stage 1, human enterprise managers use TOPI (Team-Oriented Programming Interface) to specify a team-oriented program, with an organization, its goals, and its plans. TOPI in turn passes on the program specification to Karma (stage 2). In stage 3, Karma derives the requirements for roles in the organization and searches for agents with relevant expertise (labeled "domain agents" in Figure 1). Karma can query different middle agents, an Agent Naming Service (ANS) white pages, and other directory services. Karma then aids the developer in assigning these agents to organizational roles.

Having thus fully defined a team-oriented program, Karma launches the TEAMCOREs. Each TEAMCORE wraps an individual domain agent assigned to the organization, and the teams of wrappers jointly execute the team-oriented program. While executing this program, the TEAMCOREs broadcast information among themselves via multiple broadcast nets (stage 4). TEAMCOREs also communicate with the domain agents (stage 5). Karma "eavesdrops" on the various broadcasts to monitor the progress of the teams (stage 6), and it displays this progress to the enterprise manager.

A key novelty and strength of our framework is that powerful teamwork capabilities are built into its foundations, i.e., in the TEAMCORE wrappers. These wrappers enable agents, not originally constructed as cooperative, to still plan and act together as a team. Thus, in a team formed with TEAMCORE wrappers, agents automatically cover for failed teammates, supply key information to each other, etc. This framework strongly contrasts with previous agent integration frameworks such as the Open Agent Architecture (OAA) [5], where centralized facilitators enable agents to locate each other, but do not provide teamwork capabilities. In addition, TEAMCORE's distributed approach avoids a centralized processing bottleneck, as well as eliminating any central point of failure.

3. Karma: Specifying and Monitoring Team Programs

Karma helps a enterprise manager in three tasks important in the building of an agent organization: (i) specifying a team program; (ii) locating and assigning relevant agents; (iii) monitoring and recording agent performance.

The Team-Oriented Program A enterprise manager specifies an organization of interest via a team program: (i)

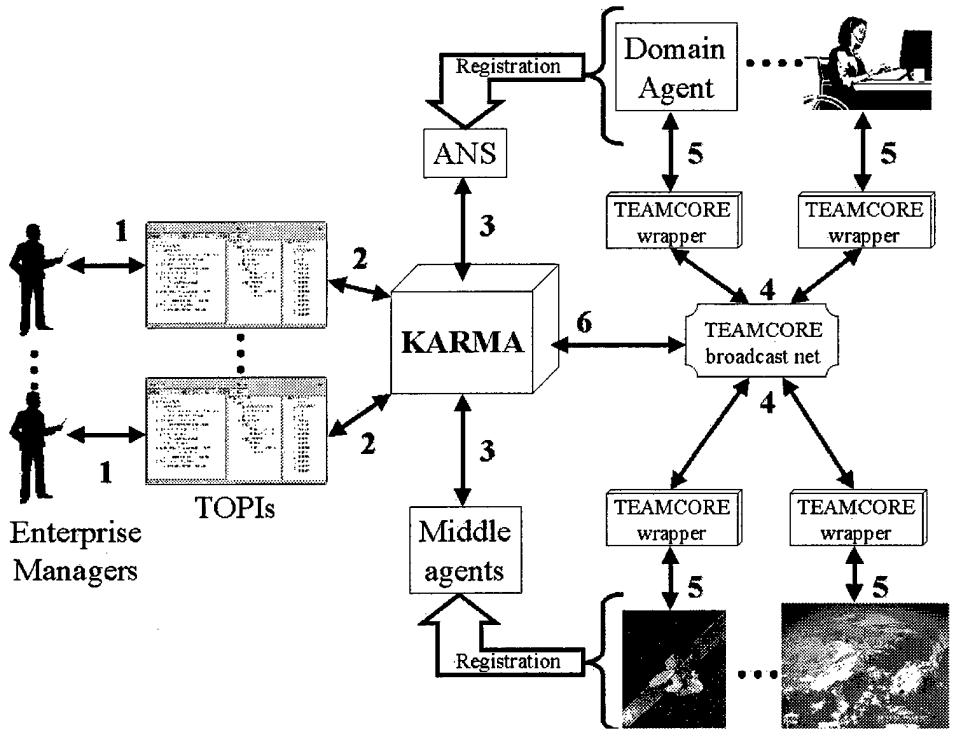


Figure 1: The overall Karma-TEAMCORE framework. TEAMCOREs may wrap different "domain agents" which may include information gathering agents, user assistants etc.

an organization hierarchy; (ii) a plan hierarchy; and (iii) assignments of agents to execute plans. The team organization hierarchy consists of roles for individuals and for groups of agents. For example, Figure 2-a illustrates a portion of the organization hierarchy involved with the evacuation scenario. Each leaf node corresponds to a role for an individual agent, while the internal nodes correspond to teams of these roles. *Task Force* is thus the highest level team in this organization, while *Orders-Obtainer* is an individual role.

The second aspect of a team program is a hierarchy of team plans explicitly expressing the joint activities of the relevant team. These plans describe the appropriate actions, the agents performing these actions, and additional high-level coordination knowledge (e.g., relevant information to be shared). The generality of the plan representation could potentially represent multi-level supply chains as the joint activity of suppliers and consumers. Figure 2-b shows an example from the evacuation scenario (please ignore the bracketed names for now). Here, high-level team plans, such as **Evacuate**, typically decompose into other team plans, such as **Process-orders**, to interpret orders provided by a human commander.

The third aspect of team-oriented programming is the assignments of agents to plans. The developer first assigns the organization's roles to plans and then assigns agents to these roles. Assigning only abstract roles rather than actual agents to plans provides a useful level of abstraction, since we can more quickly (re)assign new agents more quickly as needed. Figure 2-b shows the assignment of roles (in brackets) to the plan hierarchy for the evacuation domain. Associated with each plan is a specification of the requirements to perform the plan. A role inherits the requirements from each plan that it is assigned to.

The team program offers the key advantage of omitting the details of how to realize the specified coordination. Thus, for instance, the team designer does not program any synchronization actions — instead, during execution, the TEAMCORE wrappers automatically enforce the correct synchronization actions, both with respect to the time of plan initiation, the choice of plan, and the time of plan termination.

To facilitate the encoding of the team-oriented program, Karma interacts with a developer via the TOPI interface. Figure 3 shows a sample screenshot from programming the evacuation scenario, where the three panes correspond to the plan hierarchy (left pane), organization hierarchy (middle pane), and the domain agents (right pane). The left pane reflects the diagram 2-b. Associated with each entity are its properties, including its coordination constraints, preconditions, assigned subteam, etc.

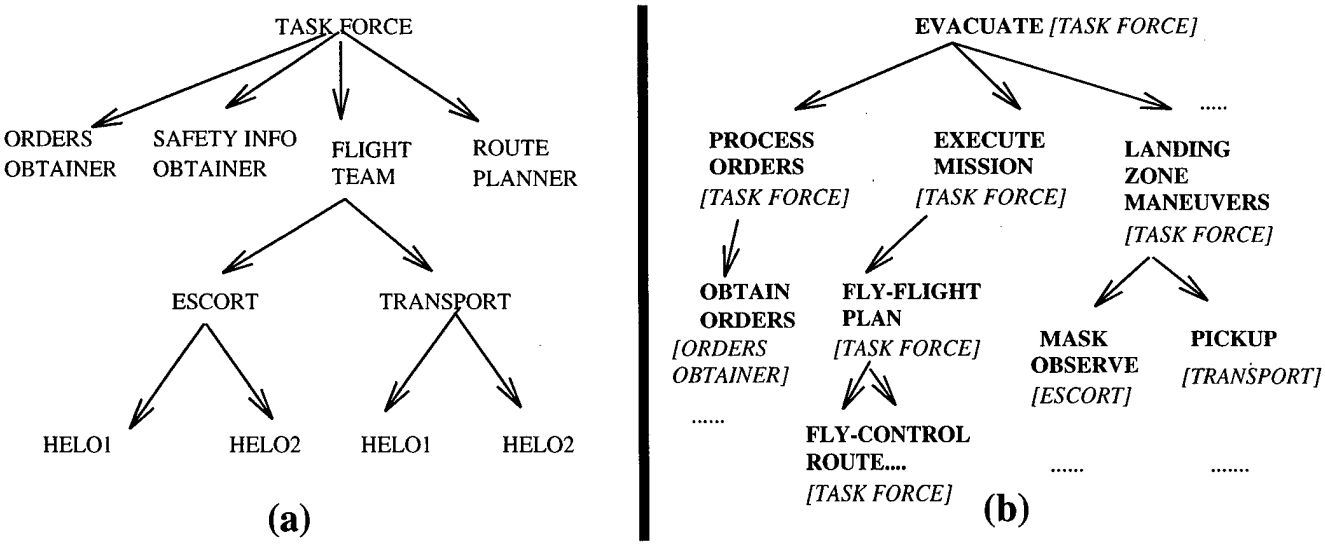


Figure 2: (a) Partial organization hierarchy with roles; (b) Partial reactive team plan hierarchy, both for the evacuation scenario.

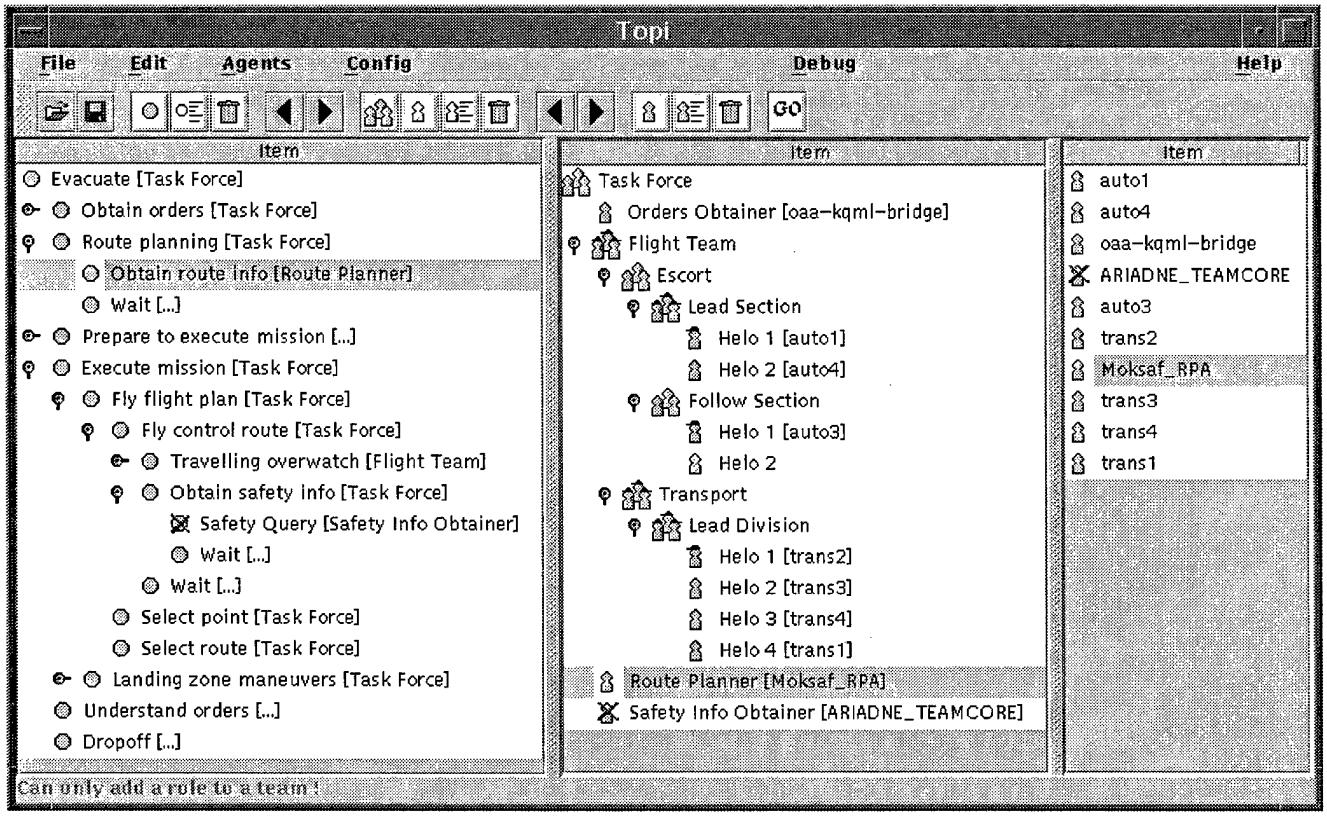


Figure 3: TOPI: The team-Oriented programming interface.

Searching and Assigning Agents Once Karma derives requirements for individual roles based on their assigned plans, it searches for agents with matching capabilities. Karma searches multiple sources: middle agents, local white pages directories of known agents, and other registry services. For instance, Karma may query the AMatchMaker [2] middle agent by sending it a KQML message specifying an advertisement template. AMatchMaker returns descriptions of those agents whose advertised capabilities match the template. In addition, Karma can search its own database of previously used agents. More recently, we have interfaced Karma with "the Grid" (produced by DARPA's COABS program [3]), an agent infrastructure providing registration and other interconnection services.

Karma compiles a list of relevant agents from these different sources, specifying their properties, including communication address, capabilities, etc. From this list, the developer can assign agents to the roles in the specified organization. By limiting its search to just those agents that meet the organizational requirements, Karma avoids overwhelming the system designer with unnecessary information.

Monitoring and Recording Agent Performance While the team executes its program, Karma's task shifts to monitoring and recording the execution. Currently, Karma's observations trigger feedback in TOPI, showing the developer which team plans and domain agents are currently active. To facilitate the reuse of agents across multiple tasks (or multiple runs of the same task), Karma also records how well each agent performs in the current task. Thus, upon completion of the task, each TEAMCORE wrapper sends Karma a report regarding the wrapped agent's performance, including any catastrophic failures during a run, success during runs, response times etc. Karma records this information in its local database. In addition, the TOPI interface provides immediate feedback for catastrophic failures to aid debugging. For instance, in Figure 3, TOPI shows the enterprise manager that Ariadne is disabled, along with its assigned role and plan.

4. TEAMCORE: Executing Team Programs

While the enterprise manager uses Karma to specify the team program and monitor its execution, the distributed set of TEAMCORE wrappers, developed in the Soar rule-based integrated agent architecture, perform the actual execution. The STEAM teamwork model [6] provides agents with three forms of domain-independent knowledge of teamwork to enable them to autonomously reason about coordination and communication. *Coherence preserving* rules require team members to communicate with each other to ensure coherent initiation and termination of team plans. *Monitor and repair* rules detect if a team task is unachievable due to unexpected member failure. It then leads the team into reorganization to overcome this failure. *Selectivity-in-communication* rules avoid excessive communication through decision-theoretic communication selectivity.

In the original STEAM implementation, the teamwork knowledge resided directly in the domain agent's knowledge base, an impractical implementation in an open, heterogeneous environment where a domain agent may be a complex legacy system. By placing this knowledge in an external TEAMCORE wrapper, we now no longer need to modify the domain agent itself. However, while the STEAM rules enable the TEAMCORE wrappers to communicate with each other automatically, we now also need a domain-agent interface module to enable a TEAMCORE wrapper to communicate requests to the domain agent it wraps. The interface module allows each TEAMCORE wrapper to send the control and information request messages that are appropriate given the current team activity. The wrapper may then communicate any response from the domain agent to the other TEAMCORE wrappers as part of the usual STEAM procedures.

We have applied our Karma-TEAMCORE framework to the mission rehearsal of the evacuation of civilians from a threatened location. The system designer created a team-oriented program for this problem, using the following agents:

Quickset: (P. Cohen et al., Oregon Graduate Institute) Multimodal command input agents [C++, Windows NT]

Route planner: (Sycara et al., Carnegie-Mellon University) Path planner for aircraft [C++, Windows NT]

Ariadne: (Minton et al., USC Information Sciences Institute) Database engine for dynamic threats [Lisp, Unix]

Helicopter pilots: (Tambe, USC Information Sciences Institute) Pilot agents for simulated helicopters [Soar, Unix]

Although none of these agents had any teamwork capabilities, we successfully used these agents within the Karma-TEAMCORE framework to build a team-oriented program for an evacuation mission rehearsal system. Karma can locate these agents based on the team-oriented program and the specified organization hierarchy. The TEAMCORE wrappers then successfully executed the team-oriented program, consisting of 18 reactive team plans. Even in the face of failures of individual agents, the entire system is robust and does not halt; instead, the team members try to substitute another agent with relevant expertise if possible and/or show graceful degradation. A second aspect of evaluation is measuring the benefit of the TEAMCORE wrappers' domain-independent teamwork knowledge, versus alternative coordination schemes. An alternative would reproduce all of TEAMCORE's capabilities via domain-specific coordination plans, where about 10 separate domain-specific

coordination plans would be needed for each team plan. In contrast, with TEAMCORE, we wrote no coordination plans for inter-TEAMCORE communication. Instead, such communications occurred automatically from the team plan specification. A third aspect of evaluation is the ease of modification to the team. For instance, the route planner was the last addition to the team. Its integration required coding of one additional role in the organization, one additional team plan in the plan hierarchy, and the specification of the new agent's capabilities. None of the existing teams, roles, or plans required any modifications.

5. Summary

Collaborative enterprises face daunting challenges when attempting the rapid integration of heterogeneous, distributed components. To this end, this article focuses on enabling designers to rapidly create agent organizations. It describes an agent resources manager, Karma, for assistance in effectively creating and managing agent organizations. As the number and variety of agents available to a particular enterprise increases, new agents, like Karma, that aid in building and maintaining agent organizations will become increasingly critical. This article also focuses on the novel TEAMCORE framework, where teamwork capabilities are built into its very foundations, through the teamwork models in our TEAMCORE wrappers. These wrappers make existing individual domain agents, who are originally not ready to be responsible team members, "team ready". Once made team-ready, these agents enable abstract specifications of an agent organization in the form of team-oriented programs. This can significantly reduce the design effort, since team-oriented programs eliminate the need to script all of the agent interactions. Our framework has shown promise, given its successful application in the concrete collaborative enterprise of the evacuation scenario.

References

[1] Mihai Barbuceanu and Mark S. Fox. The Information Agent: An infrastructure agent supporting collaborative enterprise architectures. In *Third Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises*, 1994.

[2] K. Decker, S. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-97)*, July 1997.

[3] J. Hendler and R. Metzeger. Putting it all together – the control of agent-based systems program. *IEEE Intelligent Systems and their applications*, 14, March 1999.

[4] Michael N. Huhns and Munindar P. Singh. Multiagent systems for workflow. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 8:105–117, 1999.

[5] David L. Martin, Adam J. Cheyer, and Douglas B. Moran. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1-2):92–128, 1999.

[6] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research (JAIR)*, 7:83–124, 1997.

Multi-Agent Control Strategies with Incentives

Jose B. Cruz, Jr.
Department of Electrical Engineering
The Ohio State University
Columbus, OH 43210
cruz+@osu.edu
(614) 292-1588

Marwan A. Simaan
Department of Electrical Engineering
University of Pittsburgh
Pittsburgh, PA 15261
simaan@ee.pitt.edu
(412) 624-8099

Abstract

We consider multi-agent optimization problems in which one agent is a leader and the others are followers. The leader is that agent who can declare his choice of control first. We explore the concept of control structures with incentives that the leader may wish to implement in order to induce the followers to choose their control vectors in such a way that the leader's objective function is globally optimized. Such a structure is useful in modeling future military multilevel command and control systems in intelligent hostile environments.

1. Introduction.

The design and operation of future military command and control systems in intelligent hostile environments represents a revolutionary change compared to the predominant current paradigm. The current thinking in dealing with command and control systems is rooted in single controller theory, at least in philosophy. It is possible that recent advances in single controller theory could be adapted for making significant improvements in the performance of command and control systems. However, complex military operations are processes that inherently contain at least two groups of opposing forces. Furthermore, each group may involve a team with several agents. There is a body of knowledge in control theory, called dynamic game theory, which deals with processes that are controlled by at least two sets of decision-makers. There is also another body of knowledge, called multi-agent control theory, which deals with a team of controllers. These two theories can be blended to develop a foundation for the design of multi-agent control strategies for military operation in intelligent hostile environments. In this paper we focus on developing the strategy for a team of agents, for a given strategy of the adversary. In a subsequent paper, we will address the problem of how the intelligent adversary reacts to the intelligent team strategy.

2. Multi-Agent System Dynamics.

Consider a system modeled by the difference equation:

$$x(k+1) = f[x(k), v(k), u(k), w_1(k), ..., w_N(k), z(k)]$$

where $x(k)$ is the state variable, k is discrete time, $v(k)$ is a vector of control variables at a higher level, $u(k)$ is a vector of controls for the Command and Control level, $w_1(k), ..., w_N(k)$ are vectors of control variables of peers at a lower level, and $z(k)$ is a vector of controls of an adversarial entity. In the above equation $x(k+1)$ is the state at the next discrete time, and f is a nonlinear function of the indicated variables. There is no assumption that the time instants are uniformly spaced. The actual and real times may represent discrete events. If $v(k), w_1(k), ..., w_N(k)$ and $z(k)$ are functions of $x(k)$, the above equation may be rewritten as

$$x(k+1) = g[x(k), u(k)]$$

The function "g" may appear to be rapidly changing but in fact the change may be caused by a time-varying change in $z(k)$ as a function of $x(k)$ and k in the state equation f (e.g. $z(k) = h[x(k), k]$, where h is changed to a new function as k is stepped forward). In the single controller paradigm, a command and control system considers command inputs from a higher level,

takes observations from the environment, and suggests execution orders for a Commander, to meet an objective. At a minimum, the system evaluates each option for an execution order, and ranks the options with respect to meeting the objective.

In this paper the ranking is not so simple because the effect of the external control variables must be taken into account in the state transitions. The outcome of a proposed execution order cannot be completely evaluated for the purpose of determining if the objective is met, unless the external control variables are known or intelligently guessed, particularly the adversarial control $z(k)$. In this paper, we assume that for each agent in the team, the systems that determine $v(k), u(k), w_1(k), \ldots\ldots, w_N(k)$ have completely specified objectives. The objective for each agent of the team may be the same. In general the objectives may be similar but not identical for a variety of reasons. For example the objectives for peers at a lower level may be simplified versions of the objective at a higher level, and the simplifications may be different but appropriate for each of the peers. Moreover, the levels have essential human-imbedded systems. The human elements may have slightly different perceptions of the same objectives.

The objective of the adversary is generally different from the team objective, perhaps opposite as in zero-sum dynamic games. The rationale for associating an objective for the adversary is that it is more fundamental to intelligently guess the objective than to directly guess future actions of the adversary. It is assumed that the adversary is intelligently choosing control actions to meet its objective. By using the machinery of the theory of dynamic games, the expected strategies of the adversary can be computed. For example the strategy may be $z(k) = h[x(k), k]$ where the adversary may be estimating $x(k)$ and it bases its action on the estimated $x(k)$. Estimating the objective of the adversary and then computing $h[x(k), k]$ is more reliable than guessing the strategy $h[x(k), k]$ directly. The rationale for associating separate objectives for each agent of the team is that the objectives are similar but not identical. It is assumed that each agent is intelligently choosing controls to meet its assigned objective. The process to be controlled (by a multiplicity of separate decision-makers) is more naturally investigated through the theory of dynamic games rather than through a single controller theory.

3. Incentive Strategies in Leader-Follower Games.

The concept of incentives in Leader-Follower Strategies [1-6] is a concept that we will use for the command and control hierarchy in this paper. A brief exposition of the concept of incentives in Leader-Follower Stackelberg games is included here. In order to illustrate this concept, we will consider the following simple matrix game:

		Follower Y		
Leader X		y_1	y_2	y_3
	x_1	6,4	9,4	6,3
	x_2	7,10	6,9	3,7
	x_3	9,8	5,5	1,8

The leader denoted by X has three control choices x_1, x_2, and x_3, and the follower, denoted by Y has three control choices y_1, y_2, and y_3. The entries in the ij^{th} matrix location correspond to the costs incurred for a pair of choices (x_i, y_j). The first entry is the cost incurred by the leader and the second is the cost incurred by the follower. Each player's objective is to minimize its cost, and the leader must declare his choice of control first. In order to compute its optimum Stackelberg control, the leader must first determine how the follower will react to its three possible choices.

a) If the leader chooses x_1, then the follower's optimum reaction is to choose y_3 resulting in costs of 6 for the leader and 3 for the follower.
b) If the leader chooses x_2, then the follower's optimum reaction will also be to choose y_3 resulting in costs of 3 for the leader and 7 for the follower.
c) If the leader chooses x_3, then the follower's optimum reaction is to choose y_2 resulting in costs of 5 for the leader and 5 for the follower.

Clearly, among these three possibilities the leader's best choice is x_2 resulting in a choice of y_3 by the follower and costs of 3 incurred by the leader and 7 by the follower. This solution is known as the Stackelberg

solution with player X as the leader and player Y as follower.

On the other hand, if one examines the various entries in the matrix, the optimum solution for the leader, which yields the absolute minimum cost of 1, is (x_3, y_3). However, in order for the leader to achieve this solution, it must induce the follower to choose y_3. Now, suppose instead of announcing a specific choice of control, the leader decides to announce a function $x = h(y)$ which specifies its choice of control x once the follower has chosen its control y from the set $\{y_1, y_2, y_3\}$. If this function can be chosen in such a way that the follower be induced to choose y_3, then the leader is able to achieve its objective. Such a function will then be referred to as an incentive function. Let us consider a possible such function described by:

y	$x = h(y)$
y_1	x_2
y_2	x_2
y_3	x_3

The costs incurred by the follower will be 10, 9 and 8 for the choices of y_1, y_2 and y_3 respectively. Clearly, such a function will force the follower to choose y_3. Thus, by declaring the incentive function described in the above table, the leader is able to induce the follower to choose a control that will enable it to achieve its absolute optimum solution.

2. Incentive Strategies for Multi-Agent Control.

As mentioned earlier, our main objective is to develop a framework within which the higher levels of Command and Control (C^2) use incentive controls in its interactions with the lower level agents so as to achieve optimum performance of the overall operation. To illustrate conceptually how this can be achieved, let us assume that the higher level controls are aggregated into one Command and Control level (C^2) and let us consider a simple static example that involves only one lower level control (L^2). For every possible choice of control by the adversarial entity, C^2 must decide on a strategy to optimize the performance of the overall system. Let us for simplicity assume that C^2 has a scalar decision variable x and that L^2 has a scalar decision variable y. C^2 wishes to minimize a global objective function that is influenced not only by its decision variable but also by the local decision variables of L^2. Denote this objective function by $J_C(x, y)$. Similarly, L^2 has an objective function denoted by $J_L(x, y)$. For simplicity these functions are assumed to be convex. Finally, suppose that the choices of x and y must satisfy $C(x, y) = 0$, also assumed to be convex. The relationship $C(x, y) = 0$ may represent various constraints that are imposed by the theater of operations on the variables x and y, and may also involve the choice of control by the adversary.

In general it is not possible to simultaneously minimize $J_C(x, y)$ and $J_L(x, y)$ with respect to both variables. The combination of x and y that minimizes $J_C(x, y)$ may not be the same combination that minimizes $J_L(x, y)$. Furthermore C^2 can only choose x and L^2 can only choose y. In spite of these limitations, C^2 wonders what would happen if it had complete control over the choices of both variables x and y, and if these could be chosen to minimize $J_C(x, y)$ and satisfy the constraint $C(x, y) = 0$. Suppose that the unique answer to this optimization problem is given by $x = X$, $y = Y$, and $p = P$, where p is the Lagrange multiplier in the Lagrangian function of C^2:

$$L_C(x, y, p) = J_C(x, y) + pC(x, y)$$

From C^2's perspective, optimization of the entire military operation can be achieved only if L^2 chooses, or is induced to choose, $y = Y$. Clearly, however, L^2 has no incentive to choose $y = Y$ unless it is induced to do so. With this result, C^2 wonders how it could induce L^2 to choose $y = Y$.

As illustrated in the previous section, suppose that C^2 decides to implement a sophisticated strategy whereby its decision variable x is allowed to be a function of the L^2's choice of y, i.e., $x = h(y)$, where the function h is to be determined by C^2. By doing so, C^2 is giving L^2 an incentive to influence its final choice of y. The question that still needs to be answered is: how does

C^2 select this function to induce L^2 to choose $y = Y$? In order to illustrate how this can be done, let us consider a simple example of such an incentive function. Let $x = X + A(y - Y)$, where A is a constant yet to be determined, and where Y is L^2 's control, desired by C^2, obtained as explained earlier. We will demonstrate that under some reasonable conditions this strategy will induce L^2 to choose $y = Y$. We proceed to examine L^2 's optimization problem. Knowing that x will depend on y through the above expression, L^2 proceeds to minimize its objective function $J_L(x, y)$ subject to the two constraints $x = X + A(y - Y)$ and $C(x, y) = 0$. The Lagrangian function for L^2 is

$$\begin{aligned} L_L(x, y, p_1, p_2) &= J_L(x, y) + p_1\{x - X - A(y - Y)\} \\ &\quad + p_2 C(x, y) \end{aligned}$$

where p_1 and p_2 are Lagrange multipliers. Assuming that J_L and C are differentiable, the differential of $L_L(x, y, p_1, p_2)$ can be determined as:

$$\begin{aligned} dL_L(x, y, p_1, p_2) &= \{R(x, y) + p_1\}dx \\ &\quad + \{S(x, y) - Ap_1\}dy \\ &\quad + \{x - X - A(y - Y)\}dp_1 \\ &\quad + C(x, y)dp_2 \end{aligned}$$

where

$$R(x, y) = \frac{f(J_L + p_2 C)}{fx} \quad \text{and}$$

$$S(x, y) = \frac{f(J_L + p_2 C)}{fy}.$$

Now, since $x - X - A(y - Y) = 0$, and $C(x, y) = 0$, we have:

$$\begin{aligned} dL_L(x, y, p_1, p_2) &= \{R(x, y) + p_1\}dx \\ &\quad + \{S(x, y) - Ap_1\}dy \end{aligned}$$

C^2 calculates the values of $R(x, y)$ and $S(x, y)$ at $x = X$, $y = Y$, and $p_2 = P$. If $R \neq 0$, C^2 chooses $A = -S/R$. With this choice of A by C^2, the resulting differential of the L^2 's Lagrangian function evaluated at $x = X$, $y = Y$, $p_1 = -R$, and $p_2 = P$ yields zero! Since J_L and C are convex, this first order condition is sufficient to guarantee that $y = Y$ is the unique solution for the minimization of J_L subject to the constraint of $C = 0$. If $R = 0$, then C^2 's decision variable does not affect L^2 's Lagrangian function at the C^2 's desired operating point and the incentive strategy cannot induce L^2 to choose $y = Y$. Thus, except for the "generic" case where $R = 0$, the incentive strategy of C^2 is effective.

The concept of incentive strategies can be extended to situations that involve more than one agent at the lower level. However, in this case an important question that needs to be answered as a part of the optimization process is the nature of interaction among the various Lower Level (L^2 s) agents. For example, the L^2 s may choose to cooperate among themselves and implement a Pareto-type (noninferior) solution. Or, they may be in a position that does not allow them to cooperate and hence they may implement a Nash-type solution. This issue is very important from C^2 's perspective whenever it has to deal with a multitude of Lower level agents (L^2 's). As mentioned earlier, an important objective for C^2 may be to reduce the chance of the L^2 s acting independently in a non-coordinated non-cooperative fashion. The main purpose of using incentive strategies in this case would be to induce the L^2 s to agree to cooperate in order to achieve an overall optimum of the military operation.

Let us, for illustration purposes, assume that there are N L^2 s in the incentive problem discussed earlier. Let the decision variable of C^2 be a scalar as before and let $y_1, y_2, \ldots, y_N$ denote the decision variables of the L^2 s respectively. The objective function of C^2 is now $J_C(x, y_1, y_2, \ldots, y_N)$ and the objective functions of the L^2 s are $J_{Ln}(x, y_1, y_2, \ldots, y_N)$ for $n = 1, \ldots, N$. The final choice of variables must satisfy the theater constraints $C(x, y_1, y_2, \ldots, y_N) = 0$. As before, let $x = X$, $\{y_n = Y_n, n = 1, \ldots N\}$ and $p = P$ be the unique set of variables that minimize the C^2 's Lagrangian function $L_C(x, y_1, y_2, \ldots, y_N)$. It is this solution that C^2 now wishes to induce the L^2 s to choose. However, inducing a multitude of L^2 s is more difficult than inducing one L^2 ! Following an analysis similar to the one L^2 case, a possible simple incentive function in this case would be

$$x = X + \sum_{n=1}^{N} A_n(y_n - Y_n)$$

where the A_n's are constants to be determined by C^2.

Let us first consider the case where the L^2 s wish to implement a Pareto cooperative solution among themselves. This would necessitate minimizing an objective function that is a convex combination of the individual objective functions, appended with the grid constraint and the above incentive function. That is:

$$\begin{aligned} L_L(x, y_1, y_2, ..., y_N) = & \sum_{n=1}^{N} \alpha_n J_{Ln}(x, y_1, y_2, ..., y_N) \\ & + p_1\{x - X - \sum_{n=1}^{N} A_n(y_n - Y_n)\} \\ & + p_2 C(x, y_1, y_2, ..., y_N) \end{aligned}$$

In the above expression the scalars α_n must satisfy $\sum_{n=1}^{N} \alpha_n = 1$ and $\alpha_n \geq 0$. For each choice of these α_n scalars, the proper choice of the constants A_n by C^2 can be easily determined to be $A_n = -S_n / R$,

where

$$R = \frac{f(\sum_{i=0}^{N} \alpha_i J_{Li} + p_2 C)}{fx} \text{ and } S_n = \frac{f(\sum_{i=0}^{N} \alpha_i J_{Li} + p_2 C)}{fy_n}$$

.

All these expressions must be evaluated at C^2's desired solution. It is interesting to observe that in this case each A_n will be a function of the α_n scalars. By leaving the choice of these scalars to the end, C^2 has the capability of inducing the L^2 s to implement a Pareto solution of its own choosing. In other words, C^2 is able to coordinate the nature of cooperation among the L^2's in the best way that benefits the entire military operation.

The other, and less desirable, situation is when the L^2 s are in a competitive environment and end up implementing a Nash-type solution among themselves. In this case, each L^2 will pursue a strategy of protecting itself against possible deviations by the other L^2 s from the agreed upon solution. A Nash solution

$\{y_1^*, y_2^*, ..., y_N^*\}$ must satisfy the inequalities

$$J_{Ln}(x, y_1^*, ..., y_n^*, ..., y_N^*) \leq J_{Ln}(x, y_1^*, ..., y_n, ..., y_N^*)$$

for $n = 1,2,...,N$; where x is governed by the incentive function described above. The proper choice of A_n by C^2 in this case can be shown to be $A_n = -S_n / R_n$, where R_n and S_n are given by the expressions:

$$R_n = \frac{f(J_{Ln} + p_2 C)}{fx} \text{ and } S_n = \frac{f(J_{Ln} + p_2 C)}{fy_n}.$$

As before, these expressions must be evaluated at the C^2's desired optimum overall solution. Note that in this case, C^2 has less flexibility in its ability to influence the L^2's behavior. C^2 has no parameters, such as the α_n scalars, that it can manipulate to influence the final outcome. We should mention that in the case where C^2 is using incentive controls, this solution would not have much appeal for the L^2 s. One of the main advantages of the Nash solution is in its ability to protect each L^2 against possible deviations by the other L^2 s from the agreed strategy. However, C^2 can easily provide such as guarantee by properly using its incentive controls as has been demonstrated earlier. In fact, lack of coordination among the L^2 s can be reduced to a minimum.

3. Dynamic Multi-Agent Control Strategies with Incentives.

As mentioned earlier, the structure that will be considered in modeling future command and control systems will be dynamic in nature. Dynamic decision making, in this case, typically would require a mathematical model to characterize the evolution of the entire military operation as a function of time. As we did earlier, let us assume that the higher level controls are all aggregated into one Command and Control (C^2) and that there are N agents at the lower peer level (L^2). For every possible choice of control $z(k)$ by the adversarial entity, the difference equation described in section 1 can now be written as:

$$x(k+1) = f_z(x(k), u(k), w_1(k), ..., w_N(k))$$

where $u(k)$ and $w_1(t), ..., w_N(t)$ are the control variables of C^2 and the L^2 s respectively. Typical objective functions for C^2 and L^2 may be of the form:

$$J_{C^2}(u, w_1, ..., w_N) = \sum_{k=0}^{M} I_{C^2}(x, u, w_1, ..., w_N, k)$$

and

$$J_{Ln}(u, w_1, ..., w_N) = \sum_{k=0}^{M} I_{Ln}(x, u, w_1, ..., w_N, k)$$

for $n = 1, ..., N$. In the above expressions, M is the number of time steps over which the optimization is to be performed.

Following a similar analysis as in the static case, the C^2 first minimizes $J_{C^2}(u, w_1, ..., w_N)$ as if it has complete control over the choices of u and $w_1, ..., w_N$. This would be solved using standard optimal control theory. Let U and $W_1, ..., W_N$ be the unique functions that minimize $J_{C^2}(u, v_1, ..., v_N)$. Now C^2 wants to induce the L_2's to choose $w_n = W_n$ for $n = 1, ..., N$. To achieve this, C^2 will implement a strategy $u = h(w_1, ..., w_N)$ where the incentive function h is to be determined by C^2 in such a way that the minimization of $J_{Ln}(h(w_1, ..., w_N), w_1, ..., w_N)$ by the L_2's will yield $w_n = W_n$. Since the objective function for each L_2 depends on the control choices of all L_2 s, C^2 has to take into consideration the resulting interaction among all the L_2 s. As in the static case, an objective for C^2 would be to induce the L_2 s to cooperate and achieve a non-conflicting solution. An example of a simple incentive function h is $u = U + \sum_{n=1}^{N} A_n(w_n - W_n)$ where $A_1, ..., A_N$ are appropriate functions to be determined by C^2. In the dynamic case, however, open-loop and feedback strategies need to be considered and the functions A_n will be different in each of these cases. In the open loop case, the controls u and $w_1, ..., w_N$, and the functions A_n will all be functions of time only, whereby in the feedback case, these will be functions of time and the state $x(t)$.

4. Concluding Remarks.

In this paper, we developed a multi-agent strategy in which the leader provides incentives for the other agents in the team to choose local strategies that optimize the global objective of the leader, for a given strategy of an adversary. In a subsequent paper, we will address the problem of how the intelligent adversary reacts to this intelligent team strategy, and how the team incorporates this reaction in its own optimization.

5. Acknowledgment.

Effort sponsored by the Defense Advanced Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), Air Force Materiel Command, USAF, under agreement number F30602-99-2-0549. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DARPA, the AFRL, or the U.S. Government.

8. References.

1. Tamer Basar, and Jose B. Cruz, Jr., Concepts and Methods in Multi-Person Coordination and Control. *Optimization and Control of Dynamic Operational Research Models*, Ed. S.G.Tzafestas. Amsterdam: North-Holland, 1982, 455-462.

2. Jose B. Cruz, Jr., Stackelberg Strategies for Hierarchical Control of Large Scale Systems. *Proc. Second Workshop on Hierarchical Control*, Warsaw, Poland, June 1978, 341-355.

3. Jose B. Cruz, Jr., Survey of Concepts in Hierarchical Decision-Making. *Proc. Fourth International Conf. on Analysis and Optimization of Systems*. Le Chesnay, France, December 1980, 384-396.

4. Marwan Simaan, and Jose B. Cruz, Jr., Additional Aspects of the Stackelberg Strategy in Nonzero-Sum Games. *Journal of Optimization Theory and Applications*, 11(6), 1973, 613-627.

5. Marwan Simaan, and Jose B. Cruz, Jr., On the Stackelberg Strategy in Nonzero-Sum Games. *Journal of Optimization Theory and Applications* 11(5), 1973, 533-555.

6. Marwan Simaan, and Jose B. Cruz, Jr., A Stackelberg Solution for Games with Many Players. *IEEE Trans. on Automatic Control*, AC-18, June 1973, 322-324.

Section 5

Enterprise Modeling & Control

Toward a Catalog of Pathological Behaviors in Complex Enterprise Control Systems

Alexander Kott
Logica Carnegie Group, Inc.
Pittsburgh, PA 15222
e-mail: akott@cgi.com

Bruce H. Krogh
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890
e-mail: krogh@ece.cmu.edu

Abstract

In this paper we collect a set of common undesirable behaviors that a designer of a complex enterprise control system must strive to prevent. These are some of the particularly salient issues that call for application of advanced analysis, synthesis and experimental tools and approaches. We also discuss some of the causal mechanisms that lead to the observable pathological behaviors.

1. Introduction

A catalog of pathological behaviors in a class of engineered artifacts such as enterprise control systems can serve a number of useful purposes. For example, it can:

a. serve as a guide for specification writers, designers and testers of a system;

b. help research and development community to define and communicate the areas of their current or future research, and their practical implications;

c. aid decision makers in assessing the behavior of the enterprise and predicting the effects of decisions and actions;

d. provide a framework for off-line evaluation of real and simulated situations and identifying likely sources of undesirable behaviors.

The authors of the paper were exposed to this need in the course of defining a large research project that addresses issues of applying control theory to large-scale multi-agent enterprises. The operators and domain experts in such systems had difficulties understanding the scope of the research and its relations to the practical needs and concerns. In effect, they were asking, "what kinds of practical challenges, relevant to our domain, will you address in your research?" The theoreticians and technologists had difficulties determining the desired scope and focus of the research. In effect, they were asking, "what kind of practical challenges and deficiencies would you like us to address?"

Both of these questions could be answered with greater precision and ease if one had a catalog of system-level malfunctions and pathological behaviors that occur in practical enterprise control systems and have control-theoretic implications. We were unable to find such a catalog. This paper is an attempt to begin the effort of developing the catalog.

2 Elements of the Problem

We consider enterprise control systems with multiple decision agents or decision-makers (DMs). To provide a concrete context for the behaviors described below, a simple two-level hierarchy is assumed with a single higher-level decision-maker (HLDM) and several lower-level decision-makers (LLDMs). For this situation, there are several factors that influence the ability of the enterprise control system to deal with contingencies and achieve goals. These factors include:

Internal Constraints. The options for action are always constrained by the limitations imposed by the available resources and other physical limitations within the enterprise

Environment. This refers to all aspects of the operating situation that are outside the specification and control of the DMs, including the agility and intelligence of the adversary (e.g., competitors in a market economy).

Distribution of Information. At any instant, all DMs have partial information concerning the true state of affairs. The quality of information available to any DM is affected by time (the rate at which information is made available), signal processing (the conversion of raw data into the delivered information), and communication channels (which introduce delays and possible distortion).

Distribution of Authority. In an enterprise control structure, authority is allocated among the DMs. This imposes constraints on the actions can be taken by each LLDM.

Distribution of Capability. Training and resources determine the capabilities of each DM. Typically, there is an attempt to have the distribution of authority reflect the distribution of capabilities.

Models of the Enterprise. Decisions in a complex enterprise require predictions of future behaviors. Such predictions necessarily depend on models (perhaps implicit) of the enterprise itself. Each DM models its own behavior as well as the anticipated behaviors of other DMs in the enterprise.

Models of the Environment (including adversaries). The ability to deal effectively with environmental interactions depends significantly on the sophistication and correctness of a DM's model of the environment. Note this is distinct from the particular information the DM may have about the current state of the environment. The model of the environment directly affects the ability of the DM to correctly estimate and predict future environmental interactions based on the available information.

Operational Procedures and Protocol. Predetermined rules of operation define how information will be communicated and confirmed throughout the enterprise. These procedures, combined with the enterprise and environment models, make it possible for individual DMs to predict to some extent how the system will behave.

3. Methodology

The potential pathological behaviors were culled from a variety of disciplines: discrete event system theory; classical and model-predictive control literature focused on chemical and power plant control; military Command and Control literature. Having collected a set of what appeared to be undesirable behavioral patterns in enterprise control system, we attempted a tentative classification of these patterns while distinguishing between "behaviors" and "causal mechanisms."

The "behavior-mechanism" pairs are illustrated with simple examples. Given that this research was sponsored by the DARPA JFACC Program, most of the examples come from the domain of military operations, especially air operations. However, in a few cases we used examples from other domains. In all cases we attempted to minimize the use of domain-specific terminology and the resemblance to any actual events.

We focus on pathologies that arise specifically because of the large, distributed nature of enterprise control systems. We use the term *pathological behavior* to describe undesirable system behaviors that occur despite the fact that the individual DMs are functioning correctly, that is, as they were designed to function. We use the term *causal mechanism* (sometimes abbreviated to simply "mechanism") to describe the kinds of sequences of events that occur within and between the elements of the system.

The remainder of the paper presents a collection of pathological behaviors and causal mechanisms described primarily via corresponding examples. A behavior type may have a number of subtypes. Leaf types of behavior are associated with one or more causal mechanisms. Each mechanism is illustrated by one or more examples.

4. Examples of Pathological Behaviors

The chart below presents our tentative and partial taxonomy of pathological behaviors and their causal mechanisms. In the remainder of this section we give examples illustrating the several causal mechanisms listed in the chart

Race conditions

The HLDM needs to be able to predict the system behavior to plan effectively. Predictability is lost when concurrent operations need to synchronize on conditions that are indeterminate under the given knowledge of the timing conditions. For example, if an agent is instructed to make a decision based on a condition that is changing at the time the agent is checking the condition, the agent's behavior will not be predictable. This situation is known as a *race condition* in the literature. Consider also a situation where two LLDMs A and B are instructed to use resource X and proceed with their respective mission if X is available. If resource X is not available, they are to instructed to abort their missions. Suppose the predicted time windows during which the LLDMs check for the availability of resource X overlap. It is then impossible to decide which LLDM proceeds and which LLDM aborts its mission, leaving the enterprise in an indeterminate state and, in practice, leading to delays in execution due to either the a negotiation process or an appeal to HLDM.

An assault on the Red task force is executed utilizing aircraft from three Blue carriers. The aircraft are launched at the limit of their range to optimize the element of surprise. However, enroute to the position of the enemy fleet, the different types of aircraft are unable to link up. The fighter aircraft re supposed to protect the torpedo bombers, who were relatively vulnerable during their attack run, from enemy defensive fighters. It happens that the torpedo bombers arrive at the target first, and because of fuel

Pathological Behaviors in Enterprise Control Systems

1. Behavior: System arrives to the goal state after the required deadline.
   - 1.1. Mechanism: Excessive hierarchical constraints
   - 1.2. Mechanism: Capacity saturation
2. Behavior: Reaching goal states are random-sometimes they are reached, sometimes not
   - 2.1. Mechanism: Race condition
3. Behavior: System is stalled indefinitely en route to the goal states.
   - 3.1. Mechanism: Deadlock
   - 3.2. Mechanism: Livelock
4. Behavior: System diverges from the goal states
   - 4.1. Mechanism: Positive feedback
   - 4.2. Mechanism: Cascading collapse
5. Behavior: System oscillates
   - 5.1. Mechanism: High gain
   - 5.2. Mechanism: Loss of synchronization
   - 5.3. Mechanism: High thresholds
   - 5.4. Mechanism: Hierarchical Inconsistency
   - 5.5. Mechanism: Inadequate upward information
   - 5.6. Mechanism: Excessive hierarchical constraints
6. Behavior: System wanders unnecessarily near the goal state--doesn't "lock on"
   - 6.1. Mechanism: Thrashing
   - 6.2. Mechanism: Hunting

shortage, cannot wait until the fighters arrive. Therefore they press on with their attack, and are all subsequently shot down doing no damage to the enemy fleet.

Excessive hierarchical constraints

A hierarchical control system may give the LLDMs too little authority, making it impossible for LLDMs to respond to unanticipated situations in a timely manner. Suppose an LLDM sees an opportunity to acquire a very advantageous position that would enable faster achievement of the goal, but it would require abandoning the pre-planned actions that will clearly lead to a much less attractive position. Nevertheless, the LLDM proceeds as planned because of the dire consequences that will result from delaying the planned action to request permission to do something else.

The Chief of a Combat Operations cell is concerned about recent examples of unauthorized actions by this subordinates. He leaves the cell location for a few hours and leaves clear instructions to follow his guidelines until his return. While the chief is gone, the intelligence sources pick up increased activity at an enemy chemical production site. Since this was just increased activity and the Chief would return in a few hours, the subordinates are unwilling to risk diverting assets at this early juncture without his permission. This lack of initiative by-passed an excellent opportunity to destroy significant stockpiles of chemical munitions in one place. A few hours later, the munitions are dispersed by the enemy, and are now much harder to find and to attack.

Capacity saturation

A system may max-out the memory, or processing, or bandwidth capacity of some of its components or links; this can happen under some transient conditions even if the system was properly sized for steady-state conditions. In particular, an HLDM may have too many subordinate LLDMs (excessive span of control), which will overwhelm him with information and requests for decisions, at least under certain transient conditions. Likewise, a DM may be required to have too much peer-to-peer coordination, which will overwhelm him with information and requests for decisions, at least under certain transient conditions. Coordination comes with heavy price.

Within a Combat Operations center, the planning process is divided into defensive and offensive operations. Since activity is relatively low and the personnel is limited, both types of operations are assigned to one chief. It happens that a short time later the center receives simultaneously significant defensive requirements involved with observed enemy air activity as well as several time-critical targets that require immediate diversion of the scheduled attack assets. With so many requests for action from both defensive and offensive duty officers, decisions and progress grind to a halt as the chief struggles to keep all inputs in proper perspective and sequence.

Deadlock - Resource contention

A classic problem in distributed systems is a deadly embrace in which there is a cycle of DMs waiting for resources held by other DMs in the same cycle. Suppose DM A requests information that must come from DM B, who in turn requests information from DM C, who finally requests information from DM A. But DM A cannot respond to DM C because it is waiting for information from DM B. The system is deadlocked. Or consider a case when DM A needs resources X and Y to accomplish his mission. DM B also needs resources X and Y to accomplish his mission. DM A acquires resource X; DM B acquires resource Y. Neither one can progress further. The system is deadlocked.

In the process of planning and scheduling certain operations, different components comprising the overall Task Force provide resources to the Commander. The Commander, in turn, provide services to meet the needs of the components. In order for the Commander's planning staff to formulate the plan of actions for a given period of time, the staff approaches each component and asks how much resources they are able to provide to the Commander. Each component is uncertain about how much of a service they will be allocated in the forthcoming plan, and therefore are unable to decide how much of their resources they are able to provide to the Commander. Ideally, everyone would like to receive the information from the other party before providing its own answer. In practice, each party makes assumptions and provides its best estimate, occasionally leading to significant errors.

Livelock -Thrashing

A non-convergent process of negotiations between decision-makers can lead to a pattern of thrashing cycles: (i) A recommends X1 to B; (ii) B responds with recommendation X2 to A; (iii) A recommends X3 to B; (iv) B recommends X4 to A; then (i) again, and the chain repeats. This situation might be referred to as *livelock*. Non-cyclical non-convergence is also possible.

In the XYZ Corporation, the Business Development group urges the development of a new product A. The Product Development department responds with a schedule. The Service and Support department argue that given the schedule, they will have to abandon the support to an existing product B. Given this constraint the Business Development changes its mind and asks for development of product C. The Product Development points out that the product would require certain service and the Service and Support then argues that in that case they would rather deal with product A... Now the Business Development returns to the idea of product A... and the cycle may continue for a long time.

Low threshold

Often the decisions of a DM are triggered by signals exceeding specified thresholds. When the threshold for a deviation signal is set too low, excessively frequent change of orders may occur.

The planning staff is preparing a plan of operations based on the guidance of their Commander. Before they can complete the plan, the Commander receives some additional information and decides to change his guidance to the planning staff. The staff begins to revise the partially completed plan. The Commander again receives additional information and is concerned that the plan currently being developed does not take into account the new situation. He again issues a changed guidance to his staff. The staff begins to work on the new plan that would comply with the changed

guidance, etc. The result is that the planning staff is thrashing, no plan is finalized, no orders are issued, and the entire operation stalls.

Positive feedback

Consider a situation where an HLDM issues command X to a LLDM. The LLDM executes the command and produces result Y. Through certain feedback channels (probably unintended), the HLDM receives signal Z which makes it to issue command 2*X to the LLDM, which in turn produces result 2*Y, etc.

The Control Cell diverts an attack mission towards a suddenly appearing target of high priority. The attack reports partial success. Two separate intelligence-gathering sensors report the target as still functional. Due to inadvertent and inevitable inaccuracy of the intelligence information, the Control Cell officers interpret the separate reports as indication of not one but two separate targets. Since this type of target was designated a critical priority, the cell diverts additional resources in order to attack both of the newly discovered targets. This anomaly could continue and targets would "multiply" until something/someone determines the overlapping reporting.

Cascading collapse

Cascading failures occur when more than one DM shifts an unusually large load to another DM, which in turn collapses passing an even larger load onto other DMs, and so on. Suppose the total load L is distributed equally among n DMs $X_1...X_n$, giving each DM a load of L/n. For robustness, each DM has a capacity to handle a load of 2L/n, giving it sufficient reserve to bear the normal load of one neighboring DM. Two DMs adjacent to DM X_j break down, giving a load of 3L/n to X_j. This causes X_j to break down, passing its load to a neighbor who in turn breaks down due to the load of 4L/n. The increasing load cascades until all of the DMs break down.

High gain

In enterprise control systems, a "high gain" occurs when the HLDM issues orders based on limited information and causes actions of LLDM that exceed desired limits. This problem is especially prevalent when the enterprise is forced to operate close to its limits (often optimal states). For example, in state A, the DM determines that path A-B-G will lead his system to the desired goal G, so he moves to B. At B, he finds that the system is actually moving on the path to undesirable state U. He attempts to take actions in order to move toward a better state N, but control outputs are saturated and system still enters state U (*overshoot, out-of-bound*). Alternatively, the HLDM receives feedback about the rate of achieving X. The rate of progress appears insufficient, and HLDM orders LLDM to take certain accelerating action. Execution of the action causes another state variable, Y, to exceed the desired limit (not predicted by the model available to HLDM).

A logistics cell chief is concerned about the level of non-precision munitions. His models indicate that current supply levels are below those planned. He is concerned that if no action is taken critical reserve levels will be reached. He orders an acceleration of this resupply effort. This order is dutifully obeyed, at the expense of a seemingly high resupply effort for precision munitions. Unfortunately, the models did not indicate that the air campaign would be entering a phase of operations relying almost exclusively on precision munitions. Several days of this situation leads to unacceptably low levels of precision munitions stocks.

Loss of synchronization

Synchronization is lost when different parts of a mission distributed to several components are not carried out in a coordinated, predictable way, e.g., because DMs are underconstrained resulting in dangerous mistakes when LLDMs act only on the information they have from their relatively local, myopic view of the world. For example, DM A sees an opportunity to take advantage of a local weakness in the adversary to assume a much better position. DM A leaves position X for position Y. At the same time, DM B decides to take an aggressive action toward position Z, assuming it can call on DM from position X if additional help is needed. By the time word reaches DM A at position Y that DM B needs help, it is too late.

A five Corp offensive is underway. The HLDM stressed the need to keep the Fire Support Coordination Lines aligned with the ground component boundaries. The 3rd Corp decides to take advantage of enemy weaknesses in its corridor and initiates its offensive. The forward movement and unscheduled change in forward lines creates an area of confusion. Air support is stopped in the area due to potential friendly forces in the area, and exposed flanks are created for the advancing units. The enemy takes advantage of the limited air support and counterattacks. The two flanking Corps have to come to the aid of the advancing Corp to save the isolated units.

High threshold

A system may have too high threshold for a deviation signal, causing the system to stay the course until it enters a danger state.

In a rapid response to a military contingency, limited SEAD (Suppression of Enemy Air Defenses) assets are being deployed to the theater. Although Intelligence had warned of a possible activity of sophisticated enemy air-defense missiles, the Commander made a conscious decision to hold back active participation of the limited SEAD assets until the activities of the enemy air defenses are fully confirmed. When the first two days of employment went without incidents of enemy air-defense actions, this SEAD employment decision seemed appropriate. The limited indications that the enemy was taking the opportunity to disperse their air-defense missiles were not given much weight. On the third day, attack packages were actively engaged by the dispersed enemy air defenses resulting in a very high loss rate of attack assets.

Hierarchical Inconsistency

The HLDM necessarily operates with models that involve aggregations and simplifications of the details. The use of such higher-level abstractions to make predictions about the enterprise and derive commands for the LLDMs lead to significant problems when there is an inconsistency with reality. This can happen, for example. when the model of the

state of the LLDM used by the HLDM ignores several details.

The Commander of ground troops is asking the Air Support Operations Center (ASOC) to send air sorties to support the operations of ground troops. ASOC has a limited capacity and can handle effectively only a certain number of sorties per day. When the capacity is exceeded, the excessive sorties (e.g., attack jets) arrive to the area and contact the ASOC. ASOC is saturated and is unable to issue timely orders to the pilots. The sorties, having exhausted the fuel limits, are compelled to dump their weapons and return home without delivering any useful effects within the Ground Commander's area. Meanwhile, the Ground Commander receives the information that the air support is insufficient, and not recognizing where the problem occurs naturally calls for yet more sorties.

In an air campaign, some commanders request precision-guided or penetrating munitions without full understanding of the differences in delivery difficulty and munitions availability, of particular bombs and guidance kits. Instead of letting an LLDM to make an expert decision on the appropriate type of weapon, the HLDM demands more penetrating bombs in a case when he really needs more accuracy and non-penetrating bombs would do a better job. The result is delayed sorties, reduced availability of penetrating assets that someone else really needs, and confusion regarding the real emphasis of those missions.

Inadequate upward information

When the time-space coverage of the information available to the HLDM is insufficient, the HLDM acquires an incorrect picture of the situation, making it impossible to plan effectively or correctly. Suppose the information provided to the HLDM is a time-sample (a "snapshot") of the environment at regular intervals. The HLDM interpolates between these samples to infer the state of the environment at all times. The state of the environment actually varies widely between the sample times, so the decisions of the HLDM result in ineffective actions by the LLDMs. This phenomenon of significant variations in signals between sampling times is referred to as *inter-sample ripple* in the control literature.

It is also not uncommon for the HLDM to be faced with conflicting information from different LLDMs. Major blackouts have been caused in power grids because the operator (HLDM) has not correctly interpreted the meaning of apparently conflicting information from distributed equipment (LLDMs). By incorrectly assuming one source is more credible than another source, actions are taken that make the situation worse.

At 4:30 p.m., Commander A asks his subordinate about the current position of the USS Enterprise Carrier Battle Group (CVBG). The subordinate reports the CVBG is in a certain location and is conducting flight operations. This response is based on the 7:30 a.m. report, and there is no indication that this information would change. In actuality, the CVBG was routinely diverted by Commander B to hold in a different location, to await the arrival of another ship and to cease the flight operations. Commander A incorrectly informs his superior that the CVBG is within flight operations range of the Gulf of X. Based on this incorrect information, the superior orders to begin an exercise.

During the daily Intelligence briefing, the Commander is informed that Air Force obtained indications of clandestine enemy activity in an undetermined area north of Happy Airbase. He is concerned because he has very limited manpower for base defense. In light of this information he tasks the support/maintenance personnel to augment base defenses. In addition, the Commander requests confirmation through Army Intel sources. Later, the Army responds that "we hold that area," and that there has not been any incidents. Based on the new information, base defense precautions roll-back to the previous level. Later an enemy force penetrates Happy Airbase and destroys a number of aircraft.

Excessive hierarchical constraints

A hierarchical control system may give the LLDMs too little authority, making it impossible for LLDMs to respond to unanticipated situations in a timely manner. Suppose an unforeseen situation arises that puts the LLDM in grave danger. To avoid disaster, it is necessary to take an action beyond the authority allocated to the LLDM. While waiting for approval to take the necessary actions, time runs out.

Fighter planes A and B are given air-defense radar suppression missions that are essential to success of the overall strike. At night, while enroute to target deep in enemy territory, fighter A acquires, identifies, and obtains weapons solution on an enemy fighter plane. In order to satisfy the restrictive Rules of Engagement, fighter A requests clearance to fire via AWACS. While fighter A awaits clearance to fire, the enemy fighter maneuvers out of weapons range and eventually off fighter A's scope. Then the enemy fighter escapes further detection and successfully attacks fighter B, fighter A's wingman.

Thrashing

Thrashing refers to repeated steps that degenerate into an endless cycle of actions that may even return the enterprise to the same state over and over. For example, in state A, the DM sees path A-B-C as most appropriate toward the goal, and so he moves to B; from that vantage, path B-C-D looks optimal, so he moves to C; from there, path C-D-A looks optimal... Either cyclical or non-cyclical non-convergent behavior is possible.

In a manufacturing enterprise the HLDM commands a ramping up of production to acquire a specified market share. The LLDMs are successful, but after a short time it is observed that the cost of maintaining the market share is unacceptable. Consequently, the HLDM tells the LLDMs to ramp down production, relinquishing market share to competitors. The desirability of the high market share becomes evident after a short time, so the HLDM commands a ramping up again of production. This is accomplished, but after a short time it is seen the costs are too high to maintain the position and a retreat is ordered ...

Hunting

Signals drifting around their setpoints but never settling down *(hunting)* are typical in many systems with

nonlinearities. The most common example is when there is some type of "sticktion" or deadzone in a control system. The command to set the position has to reach a certain threshold before anything happens. At that point it is larger than it should be (had the actuator started moving immediately), so the system moves beyond the desired setpoint and then the same thing happens in the other direction. Quantization in sensors can lead to the same phenomenon.

The Commander is concerned with a slower than desired achievement of the required tactical results. He determines that more air sorties are required to achieve the desired tactical results. He ramps up the production of sorties via more intensive use of air assets. The desired results are achieved in the short term, however, the increased tasking strains his forces to the degree that the aircraft unserviceable rate increases unacceptably. The Commander recognizes that the intensity he ordered was excessive and scales back the intensity of operations in order to allow his forces to concentrate on fixing broken jets. But the desired tactical results are no longer being achieved, and so the Commander is compelled to ramps up the intensity of operations once again.

Conclusions

This paper represents work in progress. In many cases, the mechanisms and pathological behaviors we have listed are found in small feedback control systems. When they occur in large enterprise control systems, it is much more difficult to deal with the problems because these systems cannot be modeled as easily as physical dynamic systems. Enterprise control systems are hybrid in nature; discrete events and actions interact with continuous variables. Moreover, human decision makers introduce complexity and variability that cannot be captured with simple differential and difference equations.

Nevertheless, we believe that it will be fruitful to pursue the development of a rich control-theoretic framework for modeling and analyzing enterprise control systems. This preliminary catalog of pathological behaviors demonstrates that many aspects of their dynamics resemble phenomena that have been studied and understood in the simpler contexts of dynamic systems and computing systems. By abstracting carefully the features of enterprise control systems that lend themselves to such models and analyses, new insights should emerge into how these complex systems can be better designed and controlled.

Acknowledgments

This research was sponsored by the DARPA's JFACC program. Most of the military examples in this paper were derived from suggestions offered by Jim Bortz, Larry Ferguson, Bob Stumpf, Tim Hughes and Don Sexton of the JFACC Program team.

Stabile and Agile Scheduling of Overhead ISR Assets

H. Stephen Morse
Chief Technologist, JFACC Program Office
SM&A Corporation
3701 N. Fairfax Drive Arlington, VA 22203-1714
hmorse@snap.org

Summary

We describe a novel approach for optimized scheduling of multiple overhead imaging ISR assets against targets in an area of interest. Using replicated databases and a simulated annealing local optimizer, we ensure that redundant collections are eliminated, and that late arriving time critical targets can be inserted rapidly with minimal disruption to the current estimated collection plan. Experiments were performed on a prototype to demonstrate the *agility* of the system (measured as latency for the insertion of a TCT into a re-optimized schedule) as well as the *stability* of the system (measured as a weighted correlation between successive estimated schedules).

1. Introduction

A low-earth-orbit (LEO) imaging satellite may have many potential imaging targets within its field of view as it passes over an area of interest (AOI) – typically, far more targets than it can feasibly image due (for example) to constraints on maneuverability, imaging time, desired image quality (and associated sensor characteristics), etc. If we imagine the targets as having a computable value (based on priority, collection history, and estimated collection conditions), we may assign a value to a feasible imaging schedule as the sum of the values of the targets imaged during the pass. The job of a *pass scheduler* is to produce an optimal feasible schedule – that is, a schedule that can actually be performed by the satellite (subject to all modeled vehicle and sensor constraints), and that maximizes the total value of all targets included in the schedule.

For purposes of planning in the user community, a pass scheduler may be asked to estimate a collection schedule for an upcoming pass many hours in advance. As the time approaches for the estimated schedule to be executed by the on-orbit asset, new information may become available that affects both the value and the feasibility of the estimated schedule. Three important sources of such dynamic events include: (1) updated weather forecasts (for EO assets); (2) the arrival of new imaging targets; and (3) imaging operations by earlier arriving assets. The first and second of these are easily understood, but the third requires more discussion. A given target may (and typically will) have an associated *desired collection frequency*; in the terms introduced above, its value at any point in time will depend, in part, on its *date last seen* (that is, the time at which it was most recently imaged). An otherwise high priority target, for example, may have low value for a particular pass if it has been very recently imaged – that is, if its *age* (= elapsed time since DLS) is less than its collection frequency. In estimating the value of accessible targets, therefore, a pass scheduler must estimate the date last seen *that will be current when the pass occurs*. In particular, the pass scheduler must estimate whether or not any other passes, intervening between when the estimate is published and when it is executed, will have included the target in their schedules, thereby updating the DLS and, hence, the target value.

This is generally known as the problem of reducing, or eliminating, *redundant collects*, and is easily understood as one aspect of the more general problem of *multi-pass optimization*. Here, the scheduler attempts to optimize the total value of collections across many passes spanning a considerable length of time. And, just as dynamic events affect the estimates of any particular pass within the time window, so they will affect any attempt at multi-pass optimization.

An important characteristic of pass (and multi-pass) optimized schedulers is their ability to respond rapidly to late-arriving targets. That is, time critical targets (TCTs) with high priority and perishability may enter the system dynamically. A key performance measure is the ability of the system to rapidly generate a new, optimized schedule that includes the TCT. In this paper, we will refer to the associated latency (that is, the elapsed time from when the TCT enters the system until a new, optimized schedule is published) as the system *agility*.

Based on the discussion thus far, we are to imagine a multi-pass optimized scheduler that continuously detects and responds to dynamically occurring events by computing and republishing optimized estimated single-pass schedules within some (rolling, or receding) time horizon. For example, the arrival of a TCT will cause the "current" pass schedule to re-optimize, dropping some targets out to make room for the new TCT. These targets then become available for inclusion in down-stream schedules, which must therefore re-optimize their own estimated schedules. The resulting "ripple effect" can percolate throughout the schedule, eventually (perhaps) affecting the schedule estimates for passes occurring several hours later in time. The ability to respond rapidly to a dynamic event (the arrival of a TCT) has induced a system-level effect with potentially far-reaching consequences. Conceptually, is it possible for a small perturbation (arrival of TCT) to have large effects (major changes to the estimated plan)? What we want is a system that is as insensitive as possible to such effects. That is, we would like the revised optimized schedule to be as "close" as possible (in a sense to be defined more precisely below) to its state prior to the occurrence of the event. We refer to this as system *stability* – the property that schedule re-optimization have as little impact as possible on previously published plans.

The system to be described addresses all of these issues. It provides: (1) highly optimized single pass schedules; (2) a simple multi-pass optimization ensuring elimination of redundant collects; (3) very fast response to late arriving TCTs (= good *agility*); and (4) measurably strong system-wide *stability* in the face of a highly dynamic environment.

In Section 2, we provide a formal statement of the problem. In Section 3, we describe our proposed solution. Sections 4 and 5 then (respectively) discuss agility and stability of the proposed design. Section 6 then summarizes the results, and indicates areas for further research and development.

2. Statement of the Problem

Within a bounded rectangular geographical region (area of interest = *AOI*), perhaps 20 degrees of equatorial longitude on a side, a list of positions (given as latitude/longitude) called *targets* is given. The targets are to be imaged by any of a constellation of low-earth-orbit (LEO) imaging satellites – perhaps identical, or perhaps with sensor characteristics varying from platform to platform. Associated with each target is a value function which depends both on target priority (given) and age (computed – the elapsed time since the last successful imaging operation on the target, its *date last seen*). For EO sensors, value may also be modified to reflect an estimate of cloud cover (that is, estimated *percent cloud free*). Typically, in a given pass over the AOI, a satellite will have many more targets within its field of view than it can image, due in particular to maneuverability constraints. A *pass scheduler* is an algorithm to optimize the total value of the targets imaged during a single pass over the AOI. That is, from the large set of feasible schedules (where a schedule is a set of imaging start and stop times for specific targets), select the one that maximizes the sum of the values of targets in the schedule [**1,4,8**].

We call a time interval during which a satellite has access to the AOI an *access window*. It is typical to have a separate scheduling process (= *pass scheduler*) for every access window. For (say) a 12 hour period, there will be many access windows, depending especially on the number of imaging platforms in the constellation and their orbital characteristics. Notionally, for (say) a constellation of 8 satellites, and for an AOI at 25 degrees North latitude, we might expect 20 access windows during a 12 hour period.

In addition to a "standing" list of targets, we also envisage a continual stream of new targets entering the system. Some of them, which we call *time critical targets* (TCTs), may have very demanding time constraints (perishability). That is, the target must be imaged very soon after it is received into the system, or its value is lost. The arrival of TCTs (plus the uncertainties of weather) introduce a dynamic element into the problem that prevents a one-time multi-pass optimization. If it were known in advance what the targets are, and what the weather conditions would be, then a multi-pass optimization process could be implemented and published. However, since high-value TCTs can (and will) arrive at any time, and since their insertion into the schedule for one window may (and typically will) bump targets which must then be absorbed by other down-stream windows, a "ripple effect" can occur: the arrival of a single TCT can have the effect of altering the schedules of (perhaps many) down-stream windows.

Further, since the age of the target depends both on the time at which it is imaged as well as the time it was *last* imaged, irrespective of which satellite imaged it, a pass scheduling algorithm

for a particular satellite and a particular window must be aware of which (if any) previously arriving windows may have or may plan to image the target. To support this, we imagine that a total schedule for all the access windows in the (say) twelve hour period is published and continually updated for purposes of planning. Schedule updates, which can occur frequently and at any time, will reflect the changes that result from incoming TCTs, weather, loss of an imaging satellite, etc.

In this context, we associate the concept of *agility* with the latency in responding to late-arriving TCTs – specifically, the amount of time it takes for a (perhaps currently active) window to modify its estimated schedule so as to generate and publish a revised schedule which includes the TCT. Formally, we define agility of the system as the *maximum* of the TCT insertion latencies in the sample generated by the experiment.

Turning next to *stability*, we first define a *weighted correlation metric*, C, for two schedules S_1 and S_2, as follows:

$$C(S_1, S_2) = \frac{2V(S_1 \quad S_2)}{V(S_1) + V(S_2)},$$

where V is the sum of the values of the targets in the schedule (or portion of the schedule). We see that $0 \le C \le 1$, with $C = 0$ when the schedules are disjoint, and $C = 1$ when they are identical [5].

Now, as time goes on, and in response to dynamically occurring events, a pass scheduler will generate a time-ordered sequence of schedules, $\langle S_0, S_1, \cdots, S_n \rangle$, corresponding to times $\langle t_0, t_1, \cdots, t_n \rangle$. We may then form the sequence of schedule-to-schedule weighted correlations as follows:

$$W_i = C(S_i, S_{i-1}), \quad i = 1,2,3,\cdots,n$$

It is the statistical properties of this time series $\langle W_i \rangle_{i=1}^{n}$ that captures what we mean by *stability*. Events which cause successive schedules to be dissimilar (that is, to have small correlations) are destabilizing. A stable algorithm, then, is one which is able to respond in such a way as to keep the number (or proportion) of such events as small as possible. Note that each pass window has its own separate such sequence, extending from the time it is initialized (nominally 12 hours prior to execution) until it is finalized and up-loaded for implementation by the on-orbit imaging satellite.

To reiterate, the goal is to devise a "scheduling architecture" that optimizes total value while also providing good agility and stability. That is, it must continually publish its current best estimate of its intended collection schedule for all the access windows within some (nominally 12 hour) time horizon. These estimates are continually revised (and published) to reflect changes due to newly arrived targets and TCTs, weather, health of the constellation, and other dynamically occurring events. A "good" architecture will be able to respond very rapidly to high priority events (system agility), settle rapidly into a new published plan, and have the further property that plan-to-plan variability is kept as small as possible (system stability).

3. Proposed Solution

We used the "natural" decoupling of the problem into parallel, concurrent schedulers, one for each access window [6]. We called these *pass window schedulers*. Next, we required each pass window scheduler to publish its current best estimate for its intended schedule to a commonly accessible "bulletin board." This bulletin board (or global shared memory) was implemented using standard commercial RDBMS replication techniques [2], with a nominal system-wide update latency of < 2 sec. That is, a "write" anywhere in the system would be observed by subsequent "reads" with a maximum latency of at most 2 seconds. In our intended implementation, this replication would extend across geographically dispersed sites; in our prototype, we used three independent copies of the database connected by a standard Ethernet LAN.

Using the bulletin board, later-occurring windows can check to see whether earlier occurring windows intend to include a particular target. If so, this "estimated time last seen" can be used to compute an "estimated target value." This is the means by which we address duplication of target collects. Later occurring windows have the responsibility to check the schedules of earlier ones (via the database), and to use this estimated DLS in estimating target value.

These ideas are illustrated in Figure 1. Weather and new targets (including TCTs) enter the system through a central hub, and are then

automatically distributed throughout the system via database replication. Thus, every site has its own local copy of the data, so that "read" operations complete very quickly. Experiments show that the ratio of "reads" to "writes" in this system is well over 100:1; and, in fact, the amount of network traffic is modest. The pass window schedulers reside at the local sites, which may be thought of as organizationally associated with different sensor types. The PWSs poll the database looking for triggering events, updating their estimated schedules, and posting these revisions back to their local database. Replication then automatically distributes the results to all other nodes.

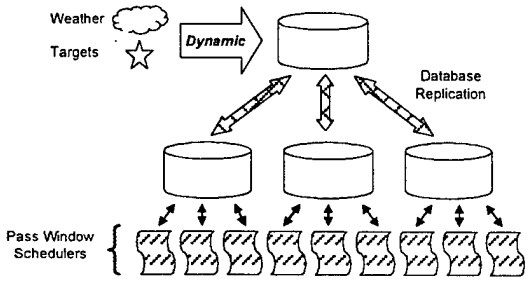


Figure 1. Top-level Architecture

While more sophisticated approaches are available, we implemented a "first-come, first served" approach to deadlock avoidance [3] or chattering. In this scheme, an *earlier occurring* PWS will never forgo taking a target because it is included in the schedule of a *later* window. This reflects our problem domain, in which we anticipate the frequent arrival of TCTs and other dynamic occurrences. While we are aware that this is sub-optimal from a global point of view, it reflects the highly dynamic natural of our problem domain. Postponing a high value opportunity, because a later access may satisfy it, will be a poor decision if circumstances intervene that preclude the later collect from occurring. Our engineering judgment, then, was *not* to allow a scheduler to look *backward* in time, to later occurring events, but only to look *forward* in time, to earlier occurring events.

4. Agility

The shared bulletin board permits a simple level of optimization by ensuring that redundant collections do not occur. To address the issue of *agility*, we used a simulated annealing algorithm specially tailored to the application [7]. The "solution space" which the algorithm is to search is the set of all feasible schedules, where a feasible schedule is an ordered set of targets which can actually be imaged, in the stated order, subject to all modeled space craft and sensor constraints. (Of these, by far the most important and computationally demanding is operation-to-operation maneuver.) The algorithm uses two state transition functions. The first, *insert*, selects a target at random (weighted by value) from the set of available, positive valued targets not in the schedule. It then creates a new schedule which contains the target, deleting lower valued targets, if necessary, to make room for the new one. In deleting targets to find room for a new one, the targets to be deleted are grouped both singly and by pairs, and the algorithm tries these entities in value order, least to greatest. Thus, the algorithm will delete a pair of lower valued targets before deleting a single higher valued target. Special data structures keep track of "gap" in the schedule, greatly accelerating the process of "sliding" to find space for the insertion.

The second state transition function is *exchange*, in which the order of a randomly selected pair of adjacent operations is switched. While this does not change the total value of the schedule, it might increase the total available "gap" time in the schedule, (as mentioned above, gap time is a quantity which the algorithm continuously computes and updates, using specially constructed data structures for this purpose).

Considerable effort was spent tuning the algorithm. A favorable operational point was found in which ten *exchange* operations were executed for each one *insert* operation. If we call a group of 1 *insert* followed by 10 *exchanges* a *basic operation*, our trials showed the following. A basic operation (implemented in C++) could be performed in about 50 msec on a 250MHz Pentium/NT. Over 95% of the floating point operations are expended in the detailed op-to-op maneuver model.

We never found a case in which the value of the schedule increased after 2000 iterations (= 100 sec) of the basic operation; and in most cases, a very high total value (within 1% of the final optimum) was reached after only 500 iterations (= 25 sec). From an *agility* point of view, however, an important point is that a new feasible solution containing a late arriving TCT can be obtained *in only one iteration*! Using the current estimated schedule as the seed, a new (but sub-optimal) schedule containing the TCT

can be generated very rapidly with a single *insert* operation. This is important operationally, because the situation can and does arise in which low-latency in generating the schedule takes priority over a high level of optimization. Further, when more time is available in the scheduling time line, the algorithm is able to utilize it to generate increasingly optimized revisions. Finally, the SA algorithm naturally decouples, permitting efficient parallelization [6].

In the prototype implementation, 20 access window schedulers, covering an 8 hour simulation time window, were all concurrently active utilizing the replicated database as the shared "bulletin board" for all data exchange and coordination. Prior to two hours before execution, a window scheduler polls the database once per minute to see whether a triggering event has occurred that requires reoptimization and the publishing of a new, revised schedule. Once the two hour threshold was passed, the scheduler polls the database 4 times per minute, up until 15 minutes before schedule execution. At that point, a trigger is set in the database to alert for the arrival of TCTs.

Since the time granularity of our event driven simulation was only 1 minute, we could not directly simulate events occurring more rapidly than that. However, off-line analysis (using computational complexity and latency estimates like those above [3,6]) showed convincingly that end-to-end system response to a late arriving TCT could be kept to less than one minute – a factor of 10 better than customer performance goals for this operation. Within that 1 minute time budget, receipt of target and schedule insert took less than 10 sec, with the remaining part of the budget allocated for detailed schedule validation and command upload. Down-stream reoptimization due to the "ripple" effect was not included in this analysis. Typically, other dynamic events intervene before a completely quiescent state is achieved, so that it is difficult to isolate the effects of a single event. As the following analysis shows, however, overall system stability was very good.

5. Stability

A major concern when we proposed this design was the stability of the published plan over time. That is, we were concerned that, in the process of re-optimization, the newly created schedules (in response to the dynamic arrival of targets, TCTs, and changes in weather) would vary greatly, schedule to schedule. This concern was based, in part, on the knowledge that some optimization algorithms are very sensitive to even small changes in the input state [3]. This "chaotic" behavior (= system sensitivity) is operationally unfortunate, since it has the potential to disrupt the planning cycles of the end users. Our hope was that an algorithm like simulated annealing, which uses its current solution as a "seed" for its next one, might dampen this effect. In addition, several *ad hoc* controls were at our disposal, including: explicit control of the temperature function; inclusion of schedule-to-schedule correlation in the objective function; and bounds on the number of iterations.

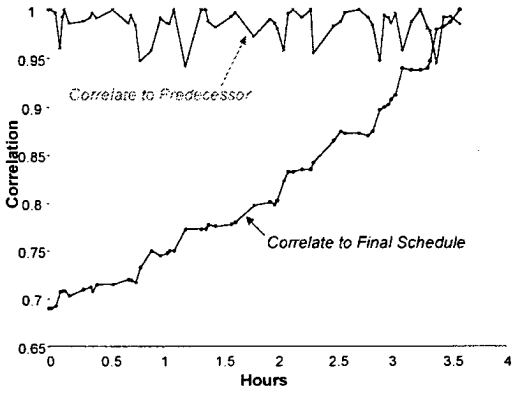


Diagram 1. Correlation Time Series

As it turns out, none of these preventive measures were necessary in order to achieve stable performance. Diagram 1 shows the results from a single PWS over a three and a half hour period prior to execution of its schedule. During that time, the PWS generated approximately 60 revised schedules. The schedule-to-schedule correlations (W_i) are plotted across the top of the chart, and we see that typical correlations are well above .95 – indicating strong schedule to schedule similarity. In a few places, the correlation is weaker. Subsequent analysis showed that, in each case, these corresponded to receipt of high-valued targets that were assigned to the pass as "mandatory" collections.

This interpretation is supported by a second graph, which plots the correlation of each new schedule to the final schedule -- the one which was uploaded to the vehicle and executed. As time proceeds, the incrementally adjusted schedules become more and more "like" the final version. This shows that the system is absorbing and adjusting to new information as it is

received; but that, on a schedule-to-schedule basis, it is only changing as much as is necessary – an excellent balance between the need for dynamic adjustment, on the one hand, and overall stability, on the other.

It should also be stressed that the simulation was taken from an extremely taxing, and highly dynamic, day in the life of an MRC. In fact, the day was chosen precisely to be the one with most new targets and TCTs. When the experiment is run on less dynamic data, the graphs show considerably less variability – remaining virtually unchanged over long periods of time. What we have shown, then, is a "worst case" example; and even here, the schedule-to-schedule stability is excellent.

What is not shown in the graph is that none of the special "damping" measures available to us were needed to achieve these results. Even using a "wide open" approach, where the initial temperature is set high and the system cools slowly, the schedule-to-schedule correlations are consistently well above .95. The large spikes occur when a high valued TCT must be inserted. In most cases, however, re-optimization is due to a "ripple" effect from a change "up stream," and the amount of change is fairly minimal. In short, the SA algorithm exhibited exactly the kind of schedule-to-schedule stability which we desired.

6. Conclusions and Issues

The work described here illustrates the following kinds of characteristics or approaches that might have some value in the JFACC program:

- use of commercial RDBMS synchronization to implement a global shared "bulletin board" for data exchange and coordination
- an SA approach with an "insert" operation able to meet tight agility latency time lines
- an SA approach with schedule-to-schedule "stability" (since new solutions are generated as deltas off of current solutions)
- a "natural" problem decomposition into local optimizations using a "look forward only" approach for deadlock avoidance

An interesting feature of this, from a JFACC perspective, is that reasonable metrics for "agility" and "stability" were generated. Perhaps these (or a suitable extension of them) might form the basis for a JFACC-program definition of these as yet ill defined notions. Another interesting feature is the use of the Linda-like "bulletin board" as the means of data exchange and co-ordination [6]. If such a capability can be implemented in a military context, it appears to offer great utility in simplifying what are currently complex push-based and "message-driven" architectures.

A weakness of the approach, however, is the use of an objective function that depends on an *a priori* definition of target value. In comparing two alternate feasible schedules, our algorithm simply sums the "given" values of the targets. This is computationally easy but operationally misleading. In fact, the actual value of one schedule vs another ought to be related (through an appropriate and perhaps complex set of intervening models) to user-defined and user-meaningful MOEs (such as attrition, *P*(success), estimated time to complete the mission, etc.). In effect, the user is forced to translate his or her heuristically meaningful MOEs into the "foreign language" of target valuation. One of the goals of the JFACC program (and other programs within DARPA) is to by-pass and/or automate that translation process. After all, why should we make life easy for the algorithmists at the expense of the war-fighter?!

References:

[1] Bate, R. R. *et al, Fundamentals of Astrodynamics* (1971, Dover, Toronto)

[2] Bobrowski, S. M., *Mastering Oracle7 & Client/Server Computing* (1995, Sybex, Alameda CA)

[3] Cormen, T. H. *et al, Introduction to Algorithms* (1994, MIT Press, Cambridge)

[4] Escobal, P. R., *Methods of Orbit Determination* (1985, Krieger, Malabar FL)

[5] Meyer, P. L., *Introductory Probability and Statistical Applications* (1970, Addison-Wesley, Reading MA)

[6] Morse, H. S., *Practical Parallel Computing* (1994, Academic Press, Cambridge)

[7] Romeo, F. et al, "A theoretical framework for Simulated Annealing," Algorithmica (1991) 6:302-345

[8] Schott, J. R., *Remote Sensing* (1997, Oxford University Press, New York)

Modeling the Requirements for Enterprise Control Systems

Philip S. Barry
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102-3481
pbarry@mitre.org

Kathryn Blackmond Laskey
Peggy S. Brouse
Department of Operations Research and Systems Engineering
George Mason University
Fairfax, VA 22032-4444
[klaskey, pbrouse]@gmu.edu

Abstract

This paper examines the use of Bayesian Networks to dynamically define and tailor the requirements for enterprise control systems. The approach taken is to model domain knowledge as Bayesian Network fragments that are glued together to form a complete view of the domain specific system requirements. Desired operational functionality is introduced as evidence and the propagation of belief is used to determine what are the appropriate system requirements. This concept has been demonstrated in the in the field of software engineering and is proposed as an effective approach for dynamic tailoring of systems.

1. Introduction

Requirements engineering consists of creating an agreement among developers, customers and users as to the intended functionality of a planned system, together with criteria for determining whether the completed system is acceptable. To reach this agreement, several distinct analysis steps must be taken. First, user requirements must be elicited. Davis [3] defines user requirements as necessary features, functions or attributes of a system that can be sensed from a position external to the system. Next, system requirements must be developed. We use Sommerville's definition of system requirements as detailed specifications of the features or functions to be implemented, together with constraints on how they are to be implemented [12]. Finally, both user requirements and system requirements are verified for completeness and consistency with each other and with user needs and domain constraints.

Increasingly, an enterprise must be concerned not only with requirements in a local domain but also with requirements that impact a local domain but are housed in different areas of the enterprise. This necessitates the integration of information throughout the enterprise. Sage defines corporate information management (CIM) as "an activity that connects humans across the organization in order to facilitate access by appropriate people, in a timely and cost effective manner, to appropriate information. This requires information access, and infrastructure to ensure pertinent information integration, and decision support for enterprise management [11]."

In general, the goals of an enterprise control system can be stated as:

- Provide assistance to managers in identifying better ways to do business (by providing standard methods and tools)
- Promote efficiencies and standardization in IT and software engineering through appropriate tools and methodsAssist in integrating common and standardized information systems within each functional area, and across functional areas of an organization
- Promote use of open systems standards
- Assist in planning for and managing development of an efficient and effective information technology infrastructure.

Mabert and Venkataramanan [10] present a hierarchical model for enterprise integration,

which links the various levels of enterprise decision processes, from strategic design planning to operational scheduling control. The model emphasizes the importance of strategic planning as the driver of an enterprises' mid- and short-term operations, as well as the necessity of an integrated information system, which they implement in a common database. Besides the required processing support, the information system also is essential in closing the feedback loop from the enterprises' daily operations back to the long-term objectives and plans. This enterprise integration model strongly reflects the concepts of vertical integration, i.e. the alignment of strategic planning and business process definition. All levels must therefore be

We extend these concepts by considering the aggregate enterprise from a systemic perspective. For each moment in time, the functionality of the system as specified by user goals can be translated into desired system functionality. Components within the enterprise provide specific functionality that can be mapped to system requirements. Thus, by specifying the goals at a given moment in time and mapping them into system requirements, the components can be tailored within the enterprise to provide the performance as indicated by the system requirements.

Agent goal statements may be at heterogeneous levels of abstraction and may engender a number of additional requirements both at the same level of abstraction as well as at higher levels of fidelity. Additionally, a priori goal prioritization schemes may result in false emphasis due to the sheer number of possible goals as well as the fact that goals are context dependent. It is clear that a scheme that maps agent goals to system requirements and then to requisite component settings must provide facilities for expressing the relationships between goals and subgoals.

The research discussed in this paper proposes that the relationships between system requirements in a domain can be modeled as a Bayesian Network. Specifically, fragments of Bayesian Networks are developed to model distinct aspects of the domain. For a particular stated set of agent goals, an appropriate set of network fragments is combined to form a Bayesian network for reasoning about the requirements derived from these goals. Evidence is introduced into the system in the form of user requirements. By observing the propagation of information through the network an assessment can be made as to what system requirements are implied by a given goal or subgoal. Once the system level goals are known, specific action necessary to adjust enterprise behavior at the component level can be taken.

2. A Conceptual Foundation for Use of Uncertain Reasoning in Requirements Modeling

The first step in defining the requirements for exercise control is to define a finite set of user goals for the enterprise. This process is non-trivial, as goal statements are usually in natural language. This requires the development of lingua franca for the translation of the natural language goals into a common syntax and semantics necessary for computational approaches.

Once a lingua franca has been defined for the anticipated goal set, a mechanism is needed to provide the mappings from stated user goals to requisite system (enterprise control) requirements. These system requirements can be functional, operational or maintenance system requirements. The mappings can become numerous and complex. We require a methodology to capture the relationships between the system requirements, such that if a given goal implies a system requirement, then related system requirements will also be invoked. Dorfman refers to this expansion of system requirements as allocation and flowdown [5].

When developing a new application in a new domain, the allocation and flowdown of requirements is a novel exercise and reuse of previous requirements definition exercises may not be possible. However, in an enterprise control situation the requisite behavior of the system can be effectively modeled as well as the dependencies of individual components to other components. When this is combined with a definition of the aggregate behavior, a complete requirements picture emerges. By pursuing a top-down functional allocation of system requirements to components, systemic behavior can be defined.

To design such a mechanism, we first consider the question of how to represent the relationship of system requirements to each other and to the

user requirements that engender them. To do this, we define an abstract structure of interrelated system requirements called a system requirement web (SRW). A SRW is a directed graph in which the nodes represent system requirements and the edges represent relationships between requirements that we call weak implication. We say that one node weakly implies another node if it is more likely that the requirement represented by the second node is needed if the first one is. User requirements can trigger a flow of weak implication within a SRW, providing a model for allocation and flowdown of requirements. Our research suggests that the use of the SRW architecture can generate specific system requirements for enterprise control systems.

Work within the software engineering domain has demonstrated that SRWs are easily modeled as Bayesian networks. A Bayesian network can be described as a factored joint probability distributed represented as a directed graph. The Bayesian network provides a computational architecture for computing the impact of evidence upon beliefs and a structure for representing knowledge about uncertain variables. Bayesian networks are directed graphs composed of nodes and arcs. The nodes represent the uncertain propositions and the arcs represent dependency relationships between the propositions. Conditional probabilities are used to model the strength the relationship between the propositions. As evidence is introduced, the posterior probabilities of the nodes are calculated, providing a measure of belief in the proposition represented by the node. A good discussion of Bayesian Networks can be found in [8].

The graphical structure of Bayesian Networks provides a natural construction mechanism for SRWs with a clear path for diagrammatically creating the relationships between the requirements. Besides providing a useful visual metaphor that aided in the construction of SRWs, the basic concepts of SRWs are readily mapped into the Bayesian Network formalism. Specifically, SRW nodes are identified with nodes in a Bayesian network referring to the proposition indicating whether or not the associated requirement is implied by the user's goals. The propagation of weak implication is modeled by propagating evidence within the Bayesian Network. Representing weak implication using conditional probabilities provides a numeric assessment for the relevance of the system requirement as part of the Bayesian Network representation. In other words, the higher the probability that a system requirement is implied (P(implied)), the more likely that it is really appropriate. Additionally, the probability that the system requirement is not needed was easily represented by P(not implied), where P(not implied) = 1-P(implied).

In previous work [1], it became apparent that the behavior of large SRWs was often difficult to predict and test. We therefore broke up the larger SRWs into manageable fragments that corresponded to a natural decomposition of the domain of interest. By combining these fragments various views into the domain could be created, emphasizing particular aspects that were important to the analyst. Two SRW fragments are combined by unifying nodes common to the two fragments and assigning as parents to the resulting node the union of the parent nodes in the two input SRW fragments. The conditional probabilities are subsequently reallocated. Laskey and Mahoney applied a similar approach to reasoning about military situations [9].

If the Bayesian network approach is to be feasible, a practical approach to specifying the necessary probabilities must be developed. Frequency information is unlikely to be available for estimating probabilities. Traditional knowledge elicitation is impractical due to the sheer number of conditional probabilities to be assessed. In the software engineering experiment, we obtained good results by specifying a few general rules for assigning conditional probabilities and tailoring when necessary. We hypothesize that the structure of the SRW matters more than the specific probability assessments and that this approach is extensible to large problems.

2.1 A SRW Example

Figure 1 models a SRW fragment as a Bayesian Network, and reflects the situation where Alloc (Allocation of Air Assets) and Task (Tasking of Air Assets) imply Jtask (tasking of Joint Air Assets). The node SAR (search and rescue assets) is implied by the node Jtask.

As with most analysis tasks, the example assumes that there was little enumerative data to base a frequentist assessment of conditional

probabilities. Therefore, the determination for the conditional probability tables was done heuristically. First, probabilities were selected to model that fact that both Alloc and Task are, with no other evidence, very unlikely. Setting the initial value that they are implied to 0.2 and the value that they are unimplied to 0.8 reflected this.

Next, conditional probabilities were selected to model the situation where it is likely that Jtask is implied if both Alloc and Task are implied. This was accomplished by setting the conditional probability to 0.8 if both Alloc and Task are implied and to 0.2 if neither one is implied. To provide a modicum of support if only one of the source nodes is implied, the probability that Jtask is implied if either Alloc or Task is implied was set to 0.4.

The last conditional probability table that was created was that for SAR. The modeling criteria chosen was that unless Jtask is implied it was unlikely that SAR would be implied. Defining the conditional probability that SAR was implied as 0.2 if Jtask was not implied and 0.8 if SAR was implied reflected this. These probability assessments are summarized below.

P(Alloc) = P(Task) = (0.2, 0.8) = 0.2 that the node is implied and 0.8 that the node is unimplied

P(Jtask) =

	Task Implied	Task Unimplied
Alloc Implied	(0.8, 0.2)	(0.4, 0.6)
Alloc Unimplied	(0.4, 0.6)	(0.2, 0.8)

Where given p(node) = (x, y), x is the probability that the node is implied and y is the probability that the node is unimplied.

P(SAR) =

Jtask implied	Jtask unimplied
(0.8, 0.2)	(0.2, 0.8)

With the initial conditions as indicated a calculation of the posterior distributions for the nodes in the Bayesian Network showed that the modeling assumptions above were accurately reflected. Specifically, there was a 35% probability that Jtask was implied and a 41% that SAR was implied. In other words, in the absence of any evidence there was less than a 50% probability that SAR would be implied; with no additional evidence it was believed that this requirement is not appropriate.

Suppose that concrete evidence was received that there is a need for air assets and they will be tasked. Since this is now a certainty, the Task node can be considered an observed fact, which has the effect of setting the implied probability to 1.0 and the unimplied probability to 0. This belief then propagates through the network raising the confidence that Jtask is implied to 64% and SAR to 58%. Further, if the system requirement Alloc is known the probability is raised to 80% for Jtask and 68% for SAR.

This is an opportune time to assess if any system requirements have been implied. For example, suppose that there is the belief that only system requirements that are well supported are important. To model this a search threshold of 0.75 is chosen. With this threshold, no system requirements in the example would initially be implied. Upon the introduction of the first piece of evidence, only Task would be considered an implied requirement as all other nodes had implied probabilities less than 0.75. Upon the introduction of the second piece of evidence, i.e. implying Alloc, three nodes are now implied (Task, Alloc and Jtask). However, SAR is still not implied as P(SAR=implied) = 0.68. If other evidence has been presented to indicate that this node should have been implied it may be appropriate to modify the model or alter the threshold if this answer is not considered reasonable.

If the decision is taken to modify the model, there are several routes that can be followed. First, additional nodes can be added and weak implication paths can be defined. Second, nodes can be deleted resulting in weak implication paths being redefined. Third, existing weak implication paths can be modified which essentially consists of modifying the conditional probability tables. Note that these modifications can be made via local changes to the SRW, leaving other parts of the SRW unchanged. This property is important from a configuration management standpoint.

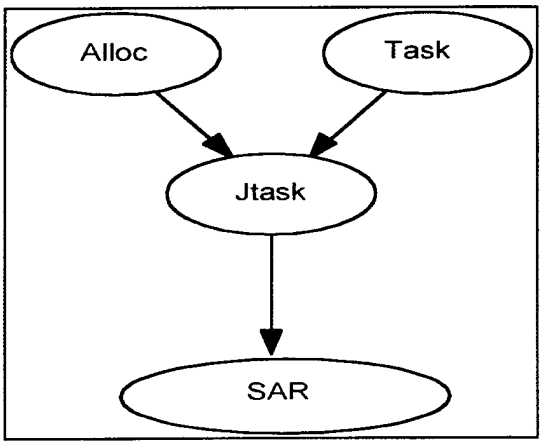


Figure 1: SRW Example

2.2 Creating SRWs for The JFACC

A top-level model of the JFACC was be created using the description provided by the Deputy Chief of Staff, Air and Space Operations Headquarters, United States Air Force [4]. The model is object based, with major actors and their associated attributes and methods. This model is shown in figure 2.

To create a SRW from this model, the nodes and arcs must be identified from the object model. First, each entity is defined by a node and a hierarchy is established between each entity. As Bayesian networks are DAGs, the relationship must have a precedence. This is done using a heuristic rule which seeks to understand the causal relationships between the entities. For example, in figure 2, the entity tasker has a method described as Develop_ATO(). This can be viewed as *causing* the existence of the entity ATO so the arc is drawn from Develop_ATO() to ATO.

After the primary entities are described, the next step is to define additional nodes from the methods in the objects. The precedence relationship is drawn from the entity to the methods. Relationships between the methods are also indicated. Lastly, relevant attributes of the entities which can be considered requirements for control are modeled as nodes. If methods are indicated, the causal arc is generally drawn from the method to the attribute. If no methods are indicated, the causal arrow is drawn from the entity to the method.

Lastly, conditional and prior probabilities must be defined. If frequency information is available, that should be used. However, in most cases, heuristic rules along with basic combinatorial operators such as "or" and "and" are necessary to be used to develop them. The translation of figure 2 into a SRW is shown in figure 3.

3. System Overview

The BOSH architecture was designed to demonstrate the use of SRWs in the translation of user requirements to system requirements in the software engineering realm. BOSH was inspired by Joint Requirements Planning Sessions where a group of experts will collaborate on defining system requirements. BOSH uses an architecture in which software agents mimic the experts and the SRW represents the experts' joint knowledge of the relationships between requirements. Conceptually the SRW can be thought of as a blackboard, a common knowledge structure for the dissemination of information between agents. A top-level view of BOSH is provided in Figure 4.

The BOSH architecture has been implemented in Java. BOSH uses a Bayesian Network representation of SRWs implemented using a freeware application programmers interface (API), JavaBayes v0.33 [2]. Individual SRWs are glued together using a tool written for specifically for this research. The translator from natural language user requirements to the system's internal representation is based on a Java version of the ELIZA program [14] and the agents are production rule systems based on JESS [6], the Java version of CLIPS [7].

SRW Agents make determinations about whether a node is likely to be implied or not implied. To do this, they use information created by other agents as well as the results of weak implication within the SRW. This information is communicated between agents using a common agent communication language. The effect of various actions such as the implication and not implication of nodes in the SRW is observed directly by the SRW Agent. SRW Agents can also elect to assert that a node in the SRW is implied, or retract the implication of a node that has been implied. When this happens, all other agents are notified of the assertion. We also allow agents to declare soft evidence

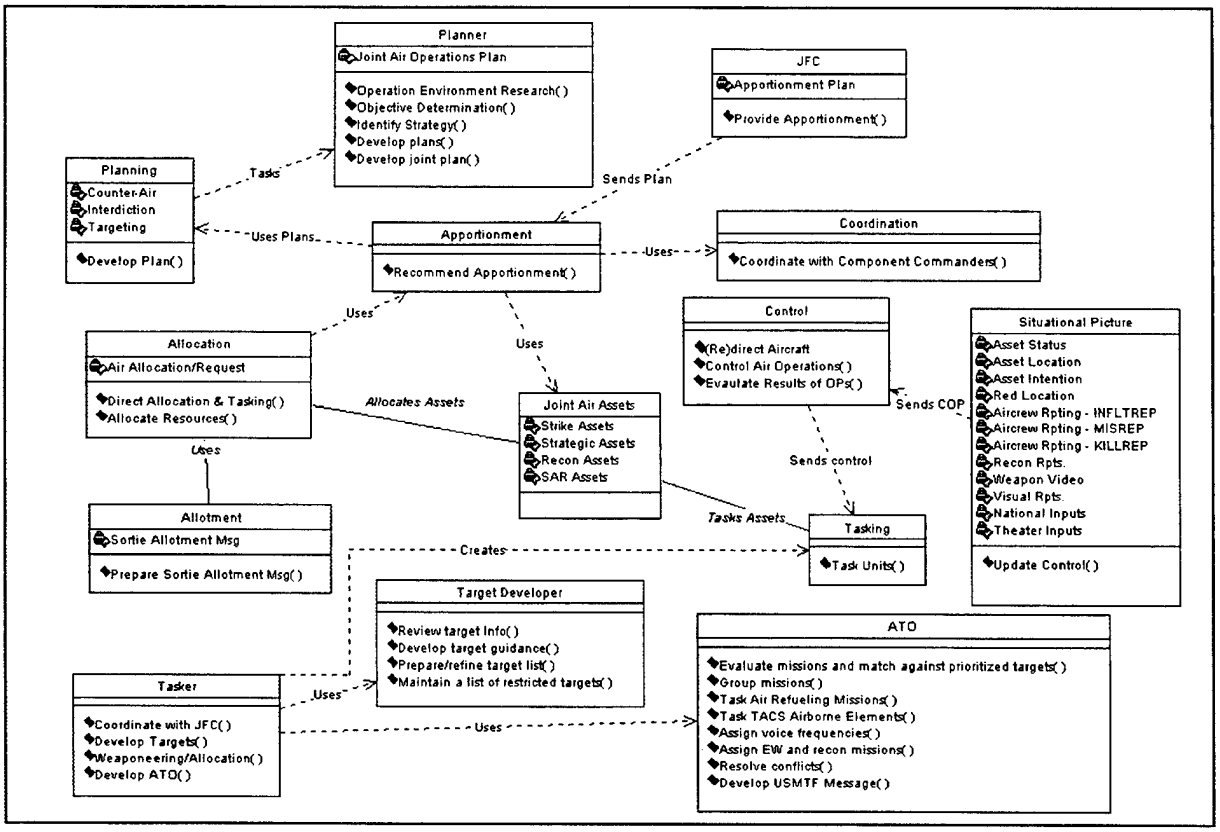


Figure 2: Top-Level JFACC Model

representing probable implication. Each agent has various goals that may be redundant or in conflict with the other agent. The normalized union of these goals at a given moment in time we consider to be the set of user requirements. This set is dynamic, changing as agents leave the enterprise, enter the enterprise or have some of their goals satisfied.

4. Extensions Required for the Enterprise Control Challenge

The primary challenge to developing a system such one described here is the creation of the initial population of Bayesian network fragments. This is similar in principle and in practice to knowledge discovery that must be done in the development of a simulation. Towards that end, the creation of a conceptual model can facilitate the description of the important entities, their interactions and the information passed between them. In a complex environment such as the JFACC, tools such as the Conceptual Model of the Mission Space [13] map textual and graphical representations of the mission space workflows and processes into a database. The mapping exercise enforces a common syntax and semantics that enables the application of analysis tools. This type of preprocessing has great utility in the development of SRWs, which are sensitive to inconsistencies and omissions.

4.1 Architectural Extensions

Compared to an enterprise control system, determining system requirements from user requirements in a software engineering venue is relatively uncomplicated. The software engineering environment functions on much longer time frames, requires decisions to be made in days or weeks. The enterprise control systems operates in real time, processing a significant amount of both external as well as internally generated information. This information can be considered evidence, introduced into the network. The evidence has associated with it various degrees of confidence. The incoming evidence also may be related to other evidence previously received.

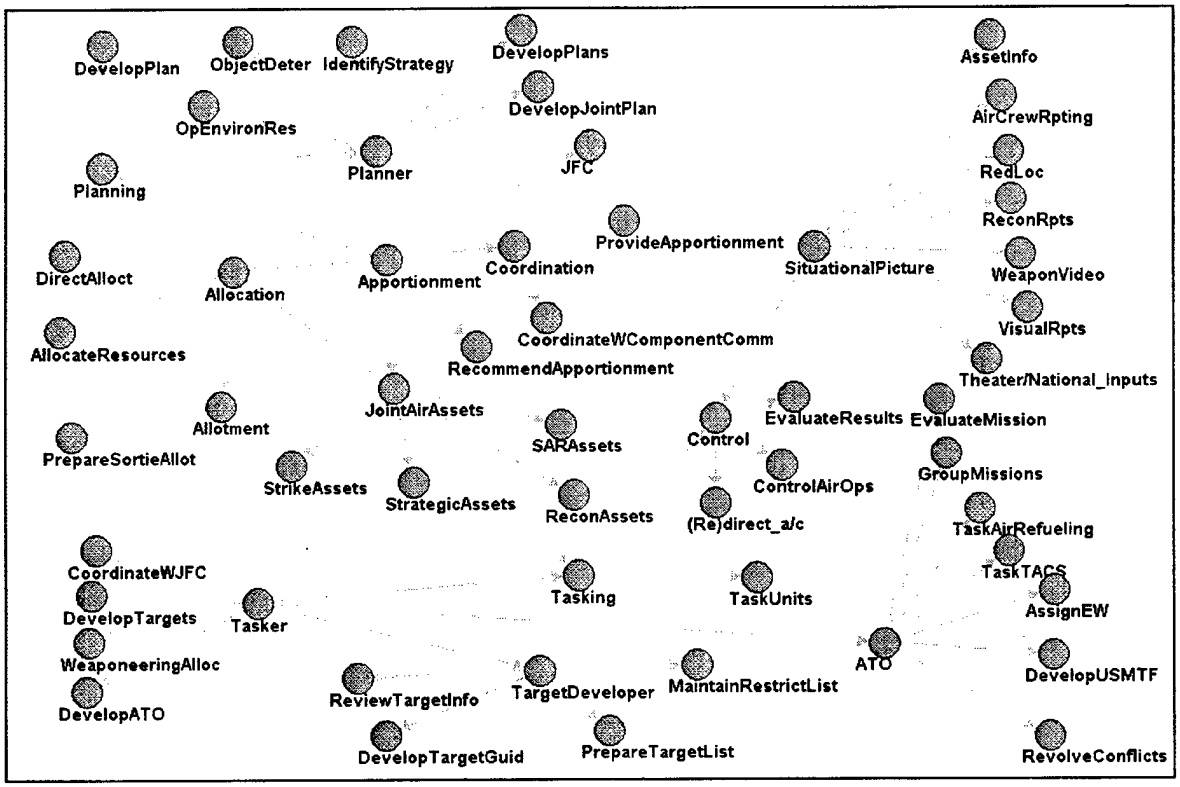


Figure 3: JFACC SRW Example

The Bayesian formalism easily accommodates the introduction of new evidence. However, the current architecture models the introduction of new evidence pre-runtime. To effectively operate within the domain of enterprise control, some mechanism must be designed to reallocate the conditional probabilities in the constructed Bayesian network. The current architecture provides the mechanical processes to introduce evidence nodes and reallocate conditional probabilities. However, the actual decision as to whether an evidence node should be introduced has been defined statically. To effectively use this architecture in a large enterprise with limited ability to define all types of incoming evidence in an a priori sense, a mechanism must be introduced to take general decisions as to what evidence is relevant.

A second area requiring additional work is in the area of context dependence. The agents in BOSH currently are rule based, with a limited understanding of the context in which they are operating. To effectively model the complex enterprise environment, the model of the agents must include a consistent operational picture as well as a modification of their internal goal structure contingent upon their operational picture. The message passing architecture of BOSH provides the underlying mechanics to communicate the goals. However, the internal modeling of the goals contingent upon the context is yet to be explored in detail.

The concept of tuning components based upon a global optimization function mapped as a system requirement has only been investigated conceptually. Technical performance measures are frequently quantified from system requirements to create verification and validation baselines as well as design to parameters in standard systems engineering approaches. However, this approach has yet to be demonstrated in a large scale dynamic tuning environment.

4.2 Development of an SRW Library

The key to using any knowledge based system is the development of a library of component pieces that will can be mined for content and used for inferencing. Using BOSH or an analogous system is no exception. It is therefore incumbent to develop SRW fragments based upon the previously mentioned conceptual analysis, available doctrine and subject matter experts.

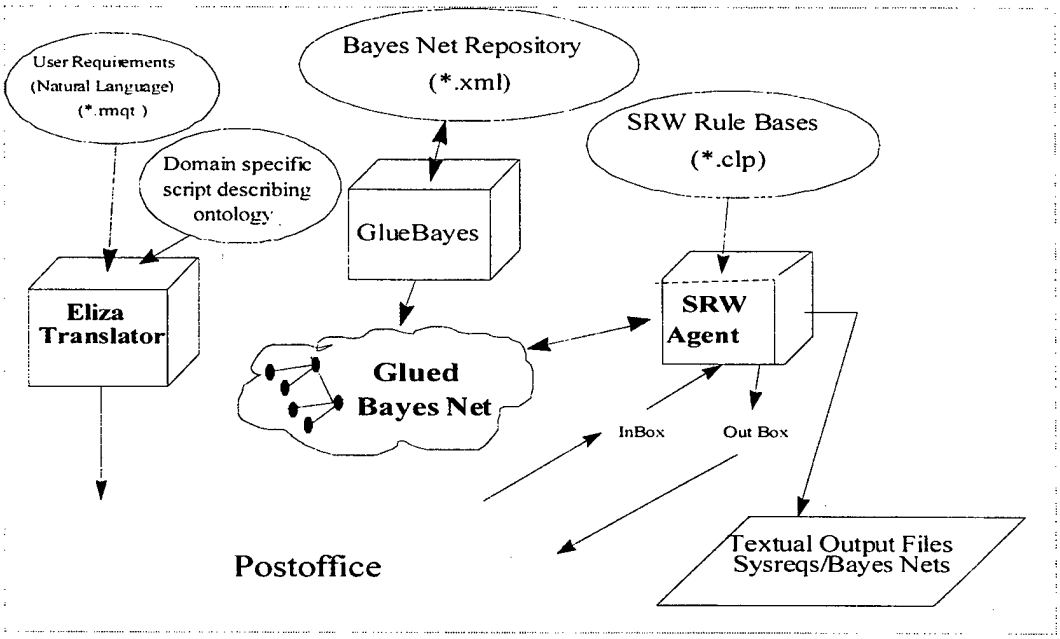


Figure 4: BOSH Architecture

The first step in developing a library of these fragments is the definition of a common syntax and semantics. The approach described herein relies on the combination of numerous fragments; to automate the combination of these fragments like named elements must have the same meaning. Additionally, the pedigree of the nodes as well as the reasons for the allocation of conditional probabilities must be fully understood. The organization of the metadata for a large system is best served by a fully indexed searchable database.

A library of SRWs requires configuration maintenance to remain current. This entails modifications to the actual nodes and their associated conditional probabilities as new information and understanding becomes available. Centrally managing the definition of nodes and providing database cataloguing of the meta-data facilitates the evolutionary use of the library.

4.3 **Development of Agents**

In BOSH, the role of the agents is to assess if incoming evidence is relevant. Relevancy is defined as information that may cause the agent to alter the conditional probabilities in the glued Bayesian network. The specific criteria that the agent uses to determine if it should alter the conditional probabilities are defined within its rule base. The rules are created from the role that the agent is designed to emulate.

In the software-engineering case study described by Barry and Laskey in [1], a hierarchical functional decomposition of the problem space was conducted. The rule bases for the agents were designed to emulate domain knowledge that roughly corresponded to these areas of functionality. In the realm of enterprise control, various decompositions can be accomplished contingent upon the view into the domain. Three decompositions are readily apparent: functional, organizational and mission based.

A functional decomposition and agent allocation is conducted by determining which type of evidence is requires specific functionality. For example, evidence may be introduced that ground forces require air cover. A functional agent can be constructed to watch for evidence related to the requirement for close air support. Upon the introduction of this evidence, the agent will create an evidence node at the appropriate place in the glued Bayesian network. Knowing where to create the evidence node is encoded in the rule base and is dependent upon a common syntax and semantics.

Organizational decompositions develop agents along the lines of the responsibilities of the organization. These agents are designed to

understand tasking that would be performed by organizational sub-elements, and create evidence nodes within the glued Bayesian network to reflect this. For example, an agent may be developed to reflect the responsibility for developing the mid-air refueling profile for the Air Tasking Order. Upon the introduction of this requirement, the agent will create the evidence node. This differs from the functional decomposition as these agents create evidence nodes along strict organizational boundaries, whereas the functional decomposition will traverse such boundaries.

A mission based agent is developed to fully understand the implications of a given mission within the domain. For example, an agent may be developed to create an ATO for reconnaissance. The agent fully understands the implications across organizational and functional boundaries for a particular mission. The agent will create evidence nodes as appropriate to reflect the responsibilities of a given resource. The resource may be across functional and organizational boundaries, such as a national asset for reconnaissance.

5.0 Summary

This paper has presented a paradigm for modeling the requirements for an enterprise control system. The problem is considered analogous to mapping user requirements to system requirements, where the goals for individual agents within the enterprise are considered as user requirements. Constructive Bayesian networks are used as the formalism to quantitatively model the relationships between system requirements. The propagation of belief within the Bayesian network models the flowdown effect based upon the introduction of agent goals.

An abstract concept called the System Requirement Web was introduced and defined as a directed graph whose nodes are connected by the weak implication operator. SRWs (instantiated as Bayesian Networks) present several attractive features. First, they are composable allowing ready construction of the requirements for the enterprise from lower level components. The method of gluing SRW fragments together is algorithmically straight forward, relying of a common syntax and semantics. Secondly, SRWs provide a multitude of views into the requirements space. The emphasis of the view into the requirements space in determined by which SRW fragments are selected. Third, the SRW construct has inherent agility. SRW fragments are easily tailored by altering conditional probabilities as well as introducing evidence. This tailoring provides fine-tuning to best represent the current understanding of the enterprise control requirements.

For large enterprise control systems consisting of a number of SRWs, a useful extension to the existing paradigm is the dynamic tailoring of SRWs predicated on some input criteria. SRWs are currently reasonably static structures, modifiable upon user intervention but only tailorable from the perspective of the propagation of belief and the creation of dynamic evidence nodes. However, it would be interesting to explore tailoring SRWs dynamically at problem solving time. This tailoring could take the form of deleting nodes that are clearly not of interest to the SRW agents which will analyze the SRW. In addition to performance increases within the prototype, a cognitive simplification could result, as networks of several hundred nodes are virtually impossible to decipher.

The SRW approach to developing technical specifications has been developed and demonstrated. The use of Bayesian Networks was shown to be an effective method for implementing an abstract concept for representing the relationships between requirements in software engineering. Future research is needed to determine the feasibility and applicability to enterprise control systems.

This architecture has been implemented to explore the translation of user requirements to system requirements in a software-engineering environment. Conceptually, there is great utility in applying this approach to the enterprise control challenge. However, there are several practical research issues that must be addressed before an operational system can be built.

References

[1] Barry, P. and Laskey, K. "An Application of Uncertain Reasoning to Requirements Engineering". Paper and Presentation for the Fifteenth Conference on Uncertainty in AI. 1999, pp. 41-48.

[2] Cozman, Fabio. JavaBayes source code and documentation available http://www.cs.cmu.edu/~javaBayes/index.html/.

[3] Davis A.M. Software Requirements Objects Functions and States. Prentice Hall. Englewood Cliffs. NJ. 1993.

[4] C2 Primer Version 9.0. Deputy Chief of Staff. Air and Space Operations Headquarters, United States Air Force, 24 Apr 97

[5] Dorfman. M. "Requirements Engineering", in **Software Requirements Engineering, 2nd Edition.** Thayer. R. and Dorfman. M. editors. IEEE Press. Los Alamitos. CA. 1997.

[6] Friedman-Hill. Ernest. JESS source code and documentation available at http://herzberg.ca.sandia.gov/

[7] Giarratano. J.C.. **CLIPS User's Guide, CLIPS Version 6.0.** Lyndon B. Johnson Space Center Information Systems Directorate Software Technology Branch. 1993.

[8] Jensen. F. **An Introduction to Bayesian Networks,** Springer-Verlag, New York, 1997.

[9] Laskey. K. and Mahoney. S. . "Network Fragments: Representing Knowledge for Constructing Probabilistic Model Networks", Paper and Presentation for the Thirteenth Conference on Uncertainty in AI. http://site.gmu.edu/~klaskey/lectures.html.

[10] Mabert. V.A., Venkataramanan, M.A., "Special Research Focus on Supply Chain Linkages: Challenges for Design and Management in the 21st Century". Decision Sciences, Vol. 29, No. 3 (1998), pp.537-552. Decision Sciences Institute.

[11] Sage. A. "Systems Reengineering" Chapter 23 in Andrew Sage and William Rouse (Editors) **Handbook of Systems Engineering and Management.** John Wiley and Sons. 1999. pp. 825-932.

[12] Sommerville I. and Sawyer P. **Requirements Engineering: A Good Practice Guide.** John Wiley and Sons Limited. West Sussex. England. 1997.

[13] Tuttle. C. et.al. "Conceptual Models of the Mission Space (CMMS) Prototype", 1997 Spring Simulation Interoperability Workshop Summary Report. Volume 1. pp. 515 - 519.

[14] Weizenbaum. Joseph. (1966) "ELIZA – A Computer Program For The Study of Natural Language Communication Between Man and Machine". Communications of the ACM no.9. pp. 36-45.

Multi-Model Predictive Control of Military Operations

Datta Godbole, Robert P. Goldman, Vipin Gopal, Jan Jelinek, Daniel P. Johnson, Blaise Morton,
Dave J. Musliner, Tariq Samad
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
Contact: jelinek_jan@htc.honeywell.com, phone (612) 951-7701

Abstract

We propose an innovative research direction for control of large scale distributed enterprises such as military operations, airline operations and distributed manufacturing. Control of such large-scale interconnected systems involves controlling individual processes as well as their interconnections. We use control of military air operations by the Joint Forces Air Component Commander (JFACC) as an example in this paper. The novel multi-model predictive control (M^2PC) method is aimed at dramatically improving agility and stability of military air operations. The M^2PC system is obtained by enhancing the models and optimization algorithms utilized in traditional Model Predictive Control systems. We describe the characteristics of the JFACC system and outline the possible performance improvement with the M2PC system.

1. Challenges of JFACC Military Air Operations

The JFACC has many responsibilities [7], which can be classified into three groups: planning and control, state estimation, and coordination with higher authorities. This proposal focuses on the planning and control aspects of the JFACC. The JFACC controls a large-scale, geographically distributed system in the presence of uncertainties and a hostile enemy (Figure 1). The JFACC control actions consist of allocating resources (wings, squadrons, air defense systems, AWACS, etc.) to different geographical locations in the theater, defining sequence of tasks for the aerospace systems at each location, and providing feedback control for the execution of these tasks. The JFACC produces an Air Tasking Order (ATO) each day that contains a sequence of tasks for each combat unit for that day [13]. A task is defined by a set point in terms of position and time where the unit should be, along with the capabilities (e.g., ammunition) of the unit and any coordination requirements with other units (e.g., air-air refueling). The tasks assigned to individual units are quite varied in nature such as bombs on targets, search and rescue, reconnaissance/surveillance, enforcing a no-fly zone, and suppression of enemy air defenses.

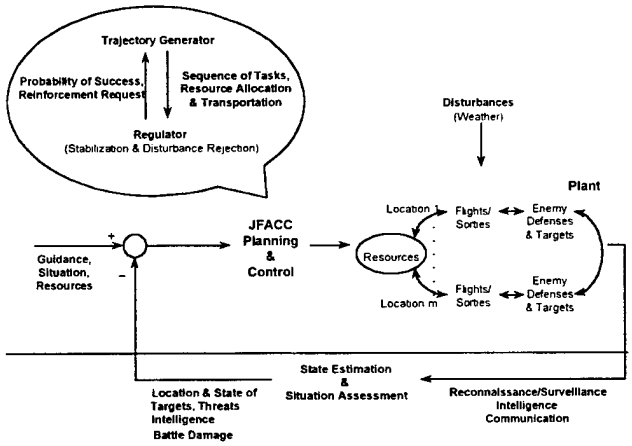


Figure 1. JFACC controls a large-scale multi-agent distributed system in the presence of uncertainties and hostile disturbances. The JFACC planning and control functionality can be represented as a hierarchical system with a trajectory generator (producing an ATO) and a regulator stabilizing the system to the desired trajectory.

Given the complexity of these tasks and the uncertainties of war, it is no surprise that changes to the plan based on real-time intelligence are frequently desired yet difficult to implement effectively. Careful planning requires the deliberation of expert teams, which takes time. For example, the planning process that produces the Air Tasking Order (ATO) takes 48 hours. The problem is that unplanned-for contingencies often arise in real time, changing the assumptions upon which the ATO is based, and requiring different actions in response. It may be difficult to determine how the ATO should best be

changed, for strategic reasons, or it may be difficult to realize the desired changes effectively for operational reasons. These are among the difficulties that we will address with control theory.

Consider the following example scenarios that may arise during execution of a JFACC plan.

1. While on a bombs-on-target or a SEAD (suppression of enemy air defenses) mission, a unit detects previously unknown enemy defensive and offensive capabilities and a major battle breaks out. Such an event requires immediate response. If the magnitude of the enemy attack is small, the contingency can be handled locally by the unit commander otherwise it needs diversion of resources from other locations by the JFACC.
2. Opening of hostilities in a new location or large resource shifts by the enemy between battle locations may require a strategic shift of friendly resources by the JFACC. The enemy actions can potentially force undesired oscillations in the JFACC resource allocations.
3. Because of the lack of precise knowledge about the success of a bombing mission, multiple flights are typically planned for a target either on the same day or on consecutive days. One of the most common contingencies is that before the second flight reaches the target, the JFACC learns of successful target destruction by the first attack. This requires real-time reallocation of resources along with on-line optimal route planning for the aircraft involved.
4. Sudden weather shifts, such as clouds or a storm moving into the theater of operation, can potentially force cancellation of several bombing missions thereby requiring the JFACC to alter its resource allocation and task assignments in real-time.

The JFACC planning and control system (see Figure 1) is responsible for computing changes to the JFACC plan and its execution based on real-time information obtained from the state estimation block. The goals of the JFACC planning and control system can be summarized as

- Drive the system to the desired end-state defined by the guidance from given authorities,
- without reaching the undesired "bad" states where enemy forces might gain the upper hand,
- while avoiding oscillations.

We will denote these goals as reachability to the end-state, while maintaining safety and stability. The performance of the JFACC controller can be measured by the time required to reach the set point and the amount of resources used (e.g., ammunition, fuel, etc.). The end-state itself is time varying as new targets and threats are constantly identified and put on the list of targets.

To effectively control military air operations in the presence of uncertainties and a hostile enemy, the JFACC planning and control function must take into account resource constraints and delays in decision making and execution. The execution delays arise due to the time it takes for the aircraft to travel from their base to the intended target location. An agile feedback system would require both reactive and proactive elements, i.e., quick response to contingencies, and ready availability of resources that can be transported quickly with minimal disruption to the overall schedule. The current system suffers from delays in planning as well as execution.

2. Solution Approach

We propose to develop Multi-Model Predictive Control (M^2PC) technology to help the JFACC planning and control system achieve agile and stable control of military operations. Our M^2PC system is based on the core technologies of MPC, hybrid systems, game theory, and probabilistic analysis using randomized algorithms. We briefly describe the benefits of each of these technologies and their application to the JFACC control problem.

The dynamics of the JFACC plant contains both continuous (positions, speeds of aircraft, time and fuel required for a particular task, etc.) and discrete (sequences of tasks in an ATO, finite resources, etc.) variables, making it a hybrid system [9]. Hybrid systems model interconnections of continuous dynamical systems and discrete event systems. Although the theories of both continuous and discrete event systems are mature, hybrid system theory is still under development. A feedback controller for the JFACC should be able to achieve reachability to the end-state with safety and stability for the hybrid model of the closed loop system. The notions of reachability, safety and stability as defined in the previous section are in accordance with the same notions used in the literature for hybrid systems.

Reachability and safety analysis is decidable for a subclass of hybrid systems called rectangular hybrid automaton [24]. The rectangular automaton will not be adequate to model the JFACC dynamics, as it cannot represent continuous dynamics more complicated than clocks with known drift. Therefore one of the main thrusts of this project is to extend the hybrid system theories so as to capture the relevant JFACC dynamics.

We use the model predictive control framework for efficient practical implementation of the hybrid system analysis and synthesis methods. Moreover, the receding horizon optimization algorithms in M^2PC will be able to numerically analyze more complicated system models than are analyzable by hybrid system theories. Even today, MPC has been successful in practice in controlling complicated nonlinear systems with time delays and constraints for which no analytical closed form feedback controller can be computed.

The MPC optimization algorithm will be replaced by game theoretic optimization in order to explicitly account for enemy actions. MPC has been used in industry to address multivariable problems on the scale of up to 100 inputs and outputs. This may still not be sufficient to handle large-scale distributed JFACC dynamical system. We will provide two solutions to the scalability issue.

We will use a hierarchical modeling and analysis method in which the lower level of the hierarchy contains detailed dynamics of each unit (or units in close geographic proximity) and the higher layer models the aggregate behavior of the entire air campaign. Given the structure of the problem, the optimization at the lower level can be performed in a distributed manner. Researchers in our team have designed hierarchical hybrid controllers for large-scale multi-agent systems in the transportation domain [1,6]. The attached paper [6] describes an application to the automated highway system design.

Secondly, we will use randomized algorithms to conduct probabilistic analysis and optimization in our predictive control setting. While the air campaign is underway, our M^{2}PC system can execute multiple runs of predictive simulations in anticipation of contingencies that may arise due to different enemy actions or weather disturbances. Randomized algorithms provide systematic ways of setting up the analysis problem, by defining the probability distributions and identifying a theoretically optimal number of predictive simulations needed for a desired level of accuracy.

Based on experience of our team in designing controllers for such large-scale applications as Automated Highway Systems and Air Traffic Management, we note that a single control technique will not be able to efficiently address the control problems in military operations. In the M^{2}PC system, we propose a set of complementary technologies that will together achieve agile and stable control of military operations.

2. 1. The M^{2}PC System

Figure 2 depicts the ways in which our M^{2}PC system helps the JFACC achieve agile and stable control of military operations.

The M^{2}PC system is assumed to receive the following inputs during an air campaign:

1. Guidance and resources coming from the higher authorities,
2. Current state of the world as assessed by the state estimation block, and
3. The current JFACC commands to the combat units.

The M^{2}PC system uses an internal hybrid dynamical model of the battle, with explicit representation of enemy actions as well as neutral disturbances such as weather, to predict the state of the campaign over a finite time horizon in the future. Game theoretic optimization algorithms produce optimal control actions that are conveyed to JFACC as suggestions. The JFACC could also use the M^{2}PC system to identify commands by running what-if experiments, particularly in order to resolve ambiguities in enemy location and intentions.

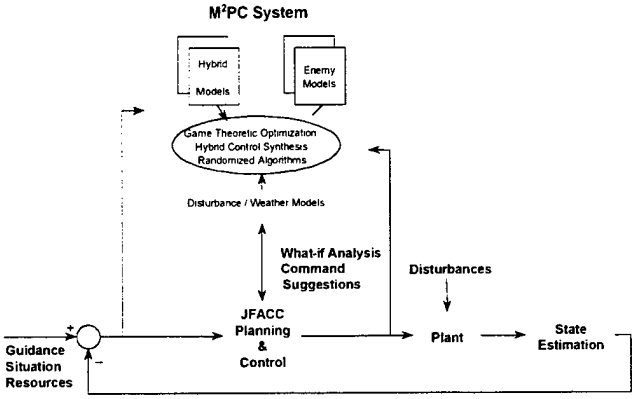


Figure 2. M^{2}PC system supports JFACC in achieving agile and stable control of military operations. The M^{2}PC system is capable of analysis as well as optimal control synthesis for the JFACC air campaign.

We propose to develop the M^{2}PC system in this project, and deliver its software implementation in a form that can be used with the enterprise models developed separately in the JFACC program. Although interaction of users (JFACC team members) with our M^{2}PC system is an important design parameter, we will not be able to devote any resources to that in this project. We will address the problems in symbolic control and hostile interactions in the uncertain environment of an air operation.

2. 2. Agility, Stability and Flexibility

Our proposed technology will dramatically improve the agility and stability of military air operations. Agility will be improved by the automation of objective choice and the rapid control system reaction to disturbances. Stability will be improved through the predictive capabilities of the MPC methodology, which allows us to choose control actions that minimize overshoot, undershoot, and oscillations. Our approach is inherently flexible (in terms of its applicability to different missions during war and peace) as it is based on an active internal model of the JFACC dynamics.

Our hierarchical modeling and analysis method within the M^{2}PC framework will achieve scalability. Because the M^{2}PC system contains multiple models (detailed models of each unit and aggregate models of the entire airspace), we can incorporate both local and global performance

criteria. In particular, disturbances resulting in small changes in resource allocation at one location should not propagate to the rest of the airspace.

We now describe the three key pillars of our proposed technology, namely model predictive control, hybrid systems and randomized algorithms. We propose to extend the existing theories and integrate them in our M^2PC system to solve the JFACC planning and control problem.

3. Model Predictive Control (MPC)

We start with a review of the current MPC technology and then describe our proposed extensions to develop the M^2PC technology.

3. 1. The MPC Concept

MPC has significantly advanced the theory and practice of control technology in the past two decades. Concepts that drive current implementations of model predictive control have been published since the 50s, and, today, as a practical advanced control scheme, MPC is the most widely used in industrial applications. Successful implementations have been reported in the petroleum refining [29], chemical [17], pulp and paper [18] and the aircraft [19] industries, among others.

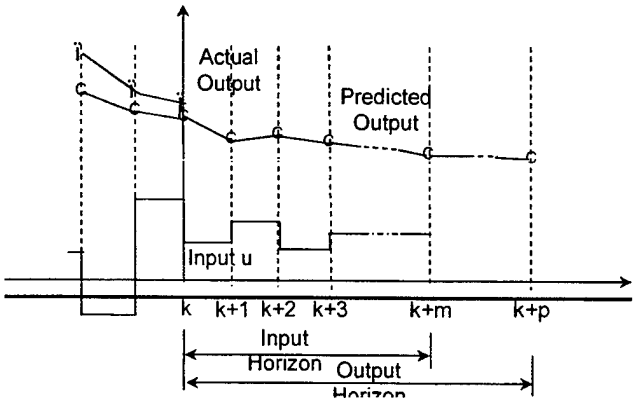


Figure 3: Receding Horizon control in MPC is effective in handling constraints and time delays

MPC is an optimal control method that uses a plant model to predict the effect of an input profile to the evolving state of a plant. At each step, an optimal control problem is solved and the optimal profile is implemented till another plant data sample becomes available. The updated plant information is used to solve a new optimal control problem and the process is repeated. This strategy yields a receding horizon control formulation as shown in Figure 3. Fundamentally one of the major advantages of the receding horizon approach is that it allows us to implement open loop strategies in a closed loop setting. This is achieved at the cost of solving an optimal control problem at each sampling instant. Recent developments in efficient optimization methods make it possible to use MPC even in a rapidly changing environment.

Four elements central to MPC make it an attractive candidate for addressing the JFACC problem:

- *Constraint handling and behavior predictions* – The immense popularity of MPC in process applications is primarily due to its explicit treatment of constraints related to the plant and its operation. Multivariable systems and their interactions are handled relatively easily through such a representation. Also, explicit prediction of future plant behavior is a feature that is an inherent component of all MPC implementations. It allows the engineer to assess the potential state of the plant in the future while making control decisions at the current time. Such properties are of prime relevance in the control of air operations.
- *Optimal corrective actions* – MPC computes corrective actions to achieve the desired output over a horizon. Typically it is posed as an optimization problem which minimizes a certain cost function subject to the relevant constraints. Fundamentally, the optimization problem formulation is generic enough and capable of handling constraints of different types. The direct incorporation of a performance criterion into the controller design is of particular importance to our problem.
- *Disturbance handling* – MPC schemes are capable of compensating for measured as well as unmeasured disturbances. Typically, the former is handled in a feed-forward fashion and the latter in a feedback fashion. Any solution to the JFACC problem should be capable of handling disturbances efficiently – MPC provides a proven way to address this issue.
- *Handling of difficult dynamics* – Industrial implementations show that MPC has achieved significant improvements over other control techniques for problems exhibiting difficult dynamics, such as long time delays. This is encouraging as well – we have to address issues including transportation delays in military operations that are conceptually similar.

Recently, significant results have appeared in the literature regarding stability and robustness analysis of MPC based implementations. The first results in the stability analysis were published in 1993 [20], for an infinite prediction horizon. The flurry of research activity since has made the nominal stability of MPC a well-understood topic. In the area of robustness, tremendous advancements have been made in the design of controllers that are robust in the presence of uncertainty in the plant

models. This has been aided by strides made in the solution of large-scale optimization problems using powerful techniques such as interior point methods [21].

The above characteristics and properties provide a solid core of fundamental technology – additional refinement and tailoring is required for application to the domain of our interest.

3. 2. M²PC for the JFACC problem

The multi-model predictive control for JFACC will be developed by adding the following capabilities to the traditional MPC framework.

- *Tackling the Hostile* – MPC has been applied in many domains in the past, but none of them have included hostile adversaries. To take the hostility element into account, we propose an approach fundamentally based on *game theory*, in which at each step of the algorithm, the optimal control problem is replaced by a dynamic game problem. In the case of continuous systems, a solution to the differential game typically requires solving a Hamilton-Jacobi partial differential equation [23]. Researchers in our group have developed hybrid controller synthesis methods based on efficient numerical algorithms to solve the Hamilton-Jacobi PDE over infinite horizon [30]. Further development of the numerical algorithms for the JFACC problem (Section 5) is likely to be needed. As the MPC implementation is based on discretized time steps, the PDE gets transformed into a difference equation, which is much easier to solve.
- *Incorporating the Hybrid* – Although most of the work in control systems research has focused on continuous dynamics, the JFACC problem is fundamentally hybrid. M²PC should be able to handle constraints in discrete-continuous form. From a theoretical perspective, the optimization problem at the heart of the MPC formulation is generic enough that it should be able to handle constraints of any type. Solution methods, such as *mixed integer linear and nonlinear programming* have recently become available [22] and are seeing continuous improvement via *interior point optimization* [21]. For hybrid systems, the execution semantics of MPC will also be changed from regular time sampling to incorporate the discrete events that are inherent in a hybrid system. The hybrid control synthesis methods developed in this project (Section 5) will also be added to the M²PC optimization suite. The synthesis methods of Section 5 provide closed form feedback solutions improving agility of response, but may not be applicable to all problems encountered in the JFACC system. On the other hand, the mixed integer optimization methods discussed above will be generally applicable.
- *Recognizing the Uncertain* – We will incorporate uncertainties in M²PC framework by making use of non-deterministic and probabilistic models. Analysis and control design will be based on randomized algorithms (Section 6) and extensions of hybrid control methods in the probabilistic setting (Section 5).

4. JFACC Models for M²PC Analysis and Synthesis

We will use multiple novel probabilistic hybrid models to capture the mixed continuous and discrete constraints of the air campaign. These models will be organized in a hierarchical manner.

4. 1. Dynamics of the Combat Units

For control design, we will model the dynamics of each agent or unit by a hybrid automaton [9]. The discrete component is given by a sequence of tasks while the continuous component describes the motion of that agent and probability of success in completing the current

A task is defined by a set point in terms of position and time where the unit should be, along with the capabilities (e.g., ammunition) of the unit and any coordination requirements with other units (e.g., air-air refueling). The models for the dynamics of "probability of success" can be adopted from the combat dynamics models in the literature [14,15,16]. The probability of success is a key variable that provides preview to the JFACC and enables him to take a proactive decision.

4. 2. Models of the Entire Air Campaign

The dynamics at the individual unit/flight level will be modeled by a network of coordinated hybrid automata. From the higher level, JFACC perspective the dynamics of individual units will be suitably abstracted as dynamical constraints in terms of resources (time, fuel, ammunition, personnel) required for a mission along with the probability of success in a given time T. This system-wide dynamics will also be modeled by a hybrid system, albeit much simpler than the model of each unit, such as a timed automaton. Our M²PC system will contain models for friendly forces, similar models for the enemy as well as models for estimating disturbances such as weather. The actual models themselves will be developed during the project with the help of domain experts provided by DARPA.

We will augment conventional hybrid control models to incorporate explicit representation of uncertainty. Currently, non-determinism is the only mechanism to represent uncertainties in a hybrid automaton. We will add probabilistic behavior to the hybrid automaton to model uncertainty in such information as the state of friendly and enemy forces, battle damage and enemy intention. The inclusion of probabilistic models allows the modeling of system components with unreliable or uncertain behavior, whereas non-determinism enables modeling of partially specified systems.

5. Formal Verification and Control Synthesis for Hybrid Systems

The research in hybrid systems has been driven by practical applications, first by timing verification in circuit designs, and then by such multi-agent systems as Automated Highways, Air Traffic Management and coordinated UAVs. The JFACC problem presents an exciting opportunity to take the theory of hybrid systems to the next level.

To date, automatic verification of hybrid systems has been limited to systems with complicated discrete dynamics but simple continuous dynamics. It has been shown that reachability computation of a hybrid system is decidable if and only if the system belongs to a restricted class, known as rectangular automaton [24]. The only continuous dynamics that can be modeled in rectangular automaton are clocks with known drift. Verification of aircraft missions however requires research and development of tools that must capture complicated nonlinear aircraft dynamics. Current state of the art tools for hybrid system verification [31] like KRONOS and HyTech do not possess such modeling expressiveness for continuous dynamics.

We propose to develop and incorporate in M^2PC framework the following hybrid systems verification and synthesis methods. These methods are designed to guarantee reachability and safety, and develop closed form feedback solutions thereby improving the speed of response. However, they may not be able to handle all the difficult dynamics posed by the JFACC problem. Therefore, these methods will be used along with the M^2PC optimization algorithms to increase agility and stability of the JFACC system.

5. 1. Hybrid System Verification Techniques

Based on very recent results from the mathematical logic of O-minimal systems, developed at U.C. Berkeley, we can verify safety properties of classes of linear hybrid systems [25]. This result enables analysis of switched linear systems, which is an important step towards modeling dynamics of each unit. Further research is needed on the complexity of these algorithms to determine whether these results can lead to a significant verification tool. The algorithms will be tested by applying them to realistic problems from military air operations pertaining to battle dynamics and mode switching at the individual unit level.

5. 2. Automatic Synthesis of Hybrid Controllers

HTC has developed the CIRCA method [10] for wholly automatic, on-line synthesis of hard real-time discrete event controllers. In this program, we will develop new automatic controller synthesis algorithms that expand the class of hybrid controllers that can be generated. Automatic controller synthesis will provide agility to the JFACC's control.

CIRCA automatically generates hard real-time controllers from a description of the platform's configuration, control objective, environment and significant environmental processes. While one of these controllers is running and operating the platform, the CIRCA controller synthesis module (CSM) generates a new controller to meet forthcoming challenges. The CIRCA CSM uses game-theoretic controller synthesis algorithms, using a real-time automaton verification tool, KRONOS, as an oracle in its search. In the interests of computational efficiency, we have restricted the set of controllers that CIRCA can synthesize.

In the JFACC program we will develop new automatic controller synthesis algorithms that can generate more capable controllers. We propose to expand the set of controllers that can be generated to cover linear hybrid automata. To do so, we will incorporate into our algorithms the more capable hybrid system verification techniques developed by Berkeley, described before, and the Hamilton-Jacobi based control synthesis methods, also developed at Berkeley, described next.

5. 3. Game Theoretic Hybrid Control Design

Researchers at Berkeley have developed a constructive methodology for the design of hybrid controllers for multi-agent systems [11]. In the continuous case, optimal controllers are derived as solutions of Hamilton-Jacobi equations (arising from differential games), while discrete controllers are synthesized by solving games on finite automata. Even though theoretically the resulting algorithms for controller synthesis are sound, their applicability is limited in practice because of the difficulty of solving the PDEs. In certain cases solutions in the conventional sense may not even exist, and one may have

to resort to weaker solution concepts, such as viscosity solutions. In addition, the class of systems for which solutions may be obtained analytically is even more limited; solutions will have to be computed numerically in most cases.

Researchers at Berkeley have been working on numerical tools for efficiently computing or approximating the solution of Hamilton-Jacobi PDEs. Our work in this project will focus on the further development of these techniques and tools. We propose to address the following computational issues:

1. Possible *methods for numerically computing solutions to PDEs* include finite element methods, level set methods, ellipsoidal or linear approximations, viability kernel computations, approximation by basis functions, and neuro-dynamic programming. We will investigate the different alternatives in an attempt to find the one best suited to the special features of the controller synthesis problem. Our investigation will be based on examples from military operations.
2. A closely related topic is the *efficient representation of the sets* involved in the controller synthesis computations. Possible choices include gridding approximations, level sets of polynomial functions, ellipsoidal approximations, etc. Each of the techniques has its advantages and disadvantages, for example, ellipsoidal approximations are amenable to numerical computation using efficient convex optimization tools, but tend to lead to significant over approximation in the case of hybrid systems. The choice of representation method will be strongly coupled with the numerical technique used to solve the PDE.

5. 4. Probabilistic Hybrid Systems

Although decidable classes of hybrid automata, such as a timed automaton, can not model aircraft and battle dynamics, they are suitable to model resource allocation at the higher level of JFACC control. For modeling military operations, we will include probabilistic models in the higher level JFACC dynamics. Our starting point has been the formulation of a probabilistic hybrid system model. We consider a probabilistic 2-sloped timed automaton whose continuous state variables are stopwatches and where an enabled transition between discrete states occurs with either zero delay or an exponentially distributed delay. The transitions from a discrete state are either with nondeterministic or probabilistic destination and, after a transition is taken, the continuous variables are either kept unchanged or reinitialized nondeterministically or probabilistically in an interval of the real numbers.

The analysis is based on the approximation of the original infinite-state system with a finite-state Markov decision process augmented by additional information on the timing behavior of the system. The verification of the properties of the Markov decision process then rests on the theory of dynamic programming and optimal control [26]. The approximation of the original system is based on an appropriate discretization procedure: as the discretization gets finer, one gets closer to the real system behavior characterization, but the required computational effort increases. The objective is to develop model-checking procedures not only for the verification of safety and reachability properties (probability that a certain behavior occurs, expected time to reach a specified set of states), but also for the assessment of the system performance in terms of long-run average properties (system throughput, average response time).

5. 5. Hybrid System Analysis using Advanced Continuous Control Methods

Even though the JFACC faces many problems that are inherently hybrid, it should be possible to simplify certain problems using results from robust control of continuous dynamical systems. Indeed most of the undecidability results of hybrid systems arise due to complicated continuous dynamics. If we could use continuous control techniques to render sets of initial conditions invariant, this information can be used to simplify hybrid control design and verification [30,11].

For example, consider the following problem at the combat unit level, represented by a state space in which the state variables are of two types:

1. the positions and velocities of aircraft, and
2. the damage expected at some future time T to each of several targets.

Here, the time T is the planning horizon of the current ATO, and, if operations proceed nominally, the probability of mission success (and level of damage expected to the targets) should increase continuously with time. If unexpected events divert one or more aircraft from their targets, or if a target is better defended than expected or moves unexpectedly, the damage expected to one or more targets may start decreasing. Depending on the relative target priorities, it may become advisable to reassign the aircraft targets—this redirection is the control input to the system.

For problems formulated with continuous-time models, there are a variety of control design and analysis techniques that can be applied, including:

1. Dynamic inversion—provides feedback control laws for nonlinear problems
2. Linearization—systems that operate close to steady state can be analyzed with linear models. When linear models are appropriate (as they may be for the example problem just described) we have more

specialized techniques, such as model reduction, H_∞ control and structural singular values.

These are state-of-the-art techniques used in design of flight control laws for modern fighter aircraft [32]. We will investigate their use in hybrid system analysis and synthesis.

6. Probabilistic System Analysis and Design Method Based on Randomized Algorithms

Randomized algorithms are proposed as computational tools for deriving quantitative and probabilistic measures of the control performance and for designing control systems optimal in some probabilistic sense.

6. 1. Randomized Algorithms

One of the especially attractive features of control science has always been its provision of "guarantees"—for a range of problems, we can devise control solutions with precisely and deterministically quantifiable properties (such as performance, stability, robustness, convergence). These comforting certainties come at a price: the problem formulations must be narrowly constrained. Thus the majority of results apply only to linear dynamical systems, objective functions must often be quadratic, disturbances are assumed to obey Gaussian distributions, etc. The deterministic setting also results in overly conservative controllers, since one single model being difficult to control forces the use of a poorly performing controller for all other models. Further, problems have proven computationally intractable. Even for linear systems, scale-up can be doubtful: determining whether the structured singular value is less than one is a certain indicator of stability, but the problem is NP-Hard and, hence, cannot be expected to be solvable for large-dimensional problems.

Fortunately, foregoing deterministic guarantees does not mean that our only recourse is to heuristics of unquantifiable efficacy. An exciting new area in control today is randomized algorithms, and it is focused on developing the tools and techniques required to produce rigorous, precise, and probabilistic measures of the quality of control solutions, and to design controllers to optimizing the average performance of the system over the uncertainty model set [27]. Such results can be obtained for general classes of problems, not limited by linearity or convexity for example, and have been applied for developing conflict detection algorithms for air traffic control [28] and designing robust adaptive controllers.

Some of the basic ideas behind randomized algorithms go back several decades—Monte Carlo simulation, Markov chain samplers, gradient-free global optimization—but it is only now that we have the processing power and memory resources to affordably tackle large-scale problems. In addition, the introduction of concepts such as VC dimension, and its extension to continuous problems via the Pollard dimension, has enabled a central shortcoming of the classical Monte Carlo approach to be overcome: problem structure can now be exploited. The numbers work out surprisingly favorably in many cases, rebutting the usual "curse of dimensionality" arguments.

As a simple example of the theory, given M independent, identically distributed samples and a desired error threshold ε, we can compute a probability measure q(M,ε) for the worst-case performance error for a control system exceeding ε as:

$$q(M,\varepsilon) \le 8\frac{16e}{\varepsilon}\ln\frac{16e}{\varepsilon}\sqrt[d]{e^{-M\varepsilon^2/32}}$$

where d is the Pollard dimension for the relevant block of the control system, and can be loosely related to the degrees of freedom in the design of the block. This result applies, with only minimal modifications, to both linear and nonlinear systems; to discrete, continuous, and hybrid dynamics; to system models, parameter and state estimators, decision and control algorithms. The two challenges are to compute d or an upper bound for it, and to draw i.i.d. samples from a complex distribution (for example, a space of controller parameters or a space of decision surfaces). Considerable research in these topics is underway, but more is needed. Notable results so far include VC dimension bounds for some types of neural network models [27] and Pollard dimension bounds for some control design methods [33].

In our M^2PC framework, instead of a closed-form calculation or a single nominal simulation at each sampling instance, a theoretically optimal number of predictive simulations (based on some extensions of the theory and formula presented above) could be undertaken for a desired level of solution confidence. These simulations are independent, and could thus be done in parallel, exploiting multi-processor PC cards or networked computing environments. For JFACC problems where analytical solutions are not possible and numerical optimization will take too long, randomized algorithms provide a revolutionary way for analysis and design of optimal response in a stochastic framework.

7. Conclusions

We have proposed a multi-model predictive control methodology that has the capability to integrate individual process optimization as well as enterprise-wide management of resources. This methodology is being developed for the control of military operations, but it can

also be applied to other large-scale enterprise systems. In the final paper, we will present a mathematical model of the JFACC system along with the controlled system responses to example disturbances characterized by enemy moves as well as weather.

Bibliography

1. C. Dhaenens-Flipo, A. Landrieu, Z. Binder, and G. Finke, "Coordination of Manufacturing and Distribution: Two Inductrial Examples", in proceedings of 14th World Congress of the IFAC, pp. 343-347, 1999.
2. K. Kosanke, and J.G. Nell, "Enterprise Organisations and the new enterprise paradigms", in proceedings of 14th World Congress of the IFAC, pp. 139-144, 1999.
3. N. Pujet, and E. feron, "Modeling an airline operations control", in proceedings of 2nd USA/Europe Air Traffic Management R&D Seminar, December 1998.
4. Jan Jelinek, "Getting HPC Performance for PID Cost", Honeywell SSDC, Jan. 93.
5. Tomlin G., J. Pappas, and S. Sastry, "Conflict resolution for Air Traffic Management: a study in multi-agent hybrid systems", IEEE Transactions on Automatic Control, Vol. 43, no. 4, pp. 509-521, 1998.
6. Lygeros J., D. N. Godbole, and S. Sastry, "Verified Hybrid Controllers for Automated Vehicles", ", IEEE Transactions on Automatic Control, Vol. 43, no. 4, pp. 522-539, 1998.
7. Joint Publication 3-56.1, "Command and Control for Air Operations".
8. Schultz, R., D. Shaner, and M. J. Hoffman, "Trajectory Optimization for the High Speed Civil Transport", Proceedings of the 1995 workshop on Trajectory Optimization Methods and Applications, NASA Conference Publication 10187, pp. 151-159.
9. Alur R., C. Courcoubetis, T. A. Henzinger, and P. H. Ho, "Hybrid Automaton: An algorithmic approach to the specification and verification of hybrid systems", in Hybrid Systems, Springer Verlag, LNCS no 736, pp. 209-229, 1993.
10. Musliner D. J., E. H. Durfee, and K. G. Shin, "CIRCA – A Cooperative Intelligent Real-Time Control Architecture", IEEE Transactions on Systems Man and Cybernetics, Vol. 23, no. 6, pp. 1561-1574, 1993.
11. Lygeros J., D. N. Godbole, and S. Sastry, "Multi-agent Hybrid System Design using Game Theory and Optimal Control", in proceedings of IEEE Conference on Decision and Control, Kobe, Japan, 1996, pp.1190-1195.
12. Plantscape, http://www.iac.honeywell.com/plantscape/index.stm
13. Major Russ Hodgkins, "Thunder Air Tasking Order Generation", Elecsim97, http://www.informatik.unibw-muenchen.de/SCS/confernc/elecsim97/papers/hodgkins/
14. M. Braun, "Differential Equations and their Applications", Springer Verlag, 1975
15. Gass, N, "Analytical model for close combat dynamics", *Journal of Operations Research Society*, Vol 48, No 2, Feb 1997, pp. 132-141.
16. Najgebauer, Andrzej; Nowicki, Tadeusz, "Modelling and analysis of conflict situations in a distributed interactive simulation environment", in Proceedings of the International Conference on Systems Science, 1998. Tech Univ Wroclaw, Wroclaw, Poland. pp. 118-125.
17. Garcia, C.E., Quadratic Dynamic Matrix Control of Nonlinear Processes-An Application to Batch Reaction Processes, *AIChE Annual Meeting,* San Francisco (1984)
18. Ricker, N.L., T. Subramaniam and T. Sim, Case Studies of Model Predictive Control in Pulp and Paper Production, in *Model Based Process Control* (Ed. T.J.McAvoy, Y. Arkun and E.Zafiriou) Pergamon (1989)
19. Mehra, R.K. *et al,* Model Algorithmic Control Using IDCOM for the F100 Jet Engine Multivariable Control Design Problem, in *Alternatives for Linear Multivariable Control* (ed. M. Sain *et al*), NEC (1978)
20. Rawlings, J.B., and K.R.Muske, The Stability of Constrained Receding Horizon Control, *IEEE Trans. Aut. Control,* 38(10): 1512-1516 (1993)
21. Gopal, V. and L.T.Biegler, Large Scale Inequality Constrained Optimization and Control, IEEE Control Systems Magazine, 18(6), 57-68 (1998)
22. A. Bemporad and M. Morari, Control of systems integrating logic, dynamics and constraints, *Automatica,* 35(3) (1999).
23. Basar T., G. J. Olsder, "Dynamic Non-cooperative Game Theory", Academic Press, 1995
24. Henzinger T., P. Kopke, A. Puri, and P. Varaiya, "What's Decidable about hybrid automata", in Proceedings of 27th Annual Symposium on Theory of Computing, STOCS'95, ACM, 1995, pp. 373-382.
25. G. Lafferriere, G. J. Pappas, and S. Sastry, "O-Minimal Hybrid Systems", To apper in Mathematics of Controls, Signals and Systems. http://robotics.eecs.berkeley.edu/~gppapas
26. L. de Alfaro, "Formal Verification of Probabilistic Systems", Ph. D. Dissertation, Department of Computer Sciences, Stanford University, 1997.
27. M. Vidyasagar (1997). *A Theory of Learning and Generalization with Applications to Neural Networks and Control Systems.* Berlin: Springer-Verlag.
28. M. Prandini, J. Lygeros, A. Nilim, S. Sastry (1999). *A probabilistic framework for aircraft conflict detection.* Proc. AIAA Guidance, Navigation and Control Conf., to appear.

29. J.M.Caldwell and J.G.Dearwater, Model Predictive Control Applied to FCC Units, *Proceedings of Chemical Process Control IV*, 319-334 (1991)
30. Lygeros, J., C. Tomlin and S.Shastry, "Controllers for Reachability Specification for Hybrid Systems", Automatica, March 99, Vol. 35, No.3. pp. 349-370
31. Henzinger, T. A., P. H. Ho and H. W. Toi, "A User Guide to HYTECH", Proceedings of TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems, Springer LNCS 1019, 1995, pp. 41 - 71
32. "Application of Multivariable Control Theory to Aircraft Control Laws", Final Report: Multivariable Control Design Guidelines, Wright Lab document WL-TR-96-3099, prepared by Honeywell Technology Center, May 1996
33. M. Prandini (1998), Adaptive Linear Quadratic Control: Optimality Analysis and Robust Controller Design, Doctoral Dissertation

Issues in the Integration of Planning and Scheduling for Enterprise Control

Karen L. Myers
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com

Stephen F. Smith
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
sfs@cs.cmu.edu

Abstract

Effective control of large-scale enterprises requires a combination of long-range planning to identify appropriate strategies for meeting objectives and efficient scheduling of resources to ensure timely completion of tasks associated with those strategies. Given the highly dynamic operating environment of many enterprises, planning and scheduling must both be adaptive to unexpected events and tightly linked to ensure responsiveness and agility. This paper discusses key issues in integrating planning and scheduling technologies to support ongoing management of enterprise operations in dynamic environments. It outlines a series of models for integrating planning and scheduling technologies, and discusses their strengths and weaknesses with respect to continuous operations management for complex enterprises. The paper also describes a new research effort that seeks to develop a dynamic integrated planning and scheduling system that can support a broad range of interactions and control strategies for the domain of air operations.

1 Introduction

The effective operation of large-scale enterprises poses significant planning and scheduling problems. Complex and frequently conflicting organizational objectives must be translated into coordinated sets of executable actions, and finite resources of the enterprise must be assigned to these actions to enable their execution in a timely and cost-effective fashion. The dynamics of the operating environment further complicate matters. As execution proceeds, unanticipated and evolving circumstances quickly and regularly force changes to strategic objectives, to planned activities, and to established resource assignments. The effectiveness of the enterprise is ultimately a function of its ability to respond to change, which in turn depends on the enterprise's ability to efficiently and appropriately adapt and manage plans and schedules over time.

One recognized obstacle to organizational responsiveness to change is poor integration of "planning" and "scheduling" processes. In manufacturing organizations, this problem has been characterized as the "wall between engineering and manufacturing". Similar sorts of barriers can be found in other large-scale enterprises. The crux of the issue is lack of communication; plans are developed with no consideration of resource availability and operational status, and likewise, schedules are developed and managed without knowledge of objectives and dependencies. Without such information exchange, reaction to unforeseen problems and opportunities necessarily proceeds in an undirected and hence inefficient manner.

A second, somewhat related obstacle to responsiveness and agility in dynamic environments is a lack of incrementality in planning and scheduling processes. In large-scale enterprises, where executing units are distributed and semi-autonomous, it is important to maintain stability and continuity in the planning and scheduling decisions that are made over time, and to minimize the impact of changes that are required to realign to current environmental circumstances. Many planning and scheduling tools/processes are not designed with incrementality in mind.

This paper considers issues related to the integration of planning and scheduling to support ongoing management of enterprise operations in a dynamic environment. By planning, we refer generally to the process of deciding *what* to do; i.e., the process of transforming strategic objectives into executable task (or activity) networks. We use the term scheduling generally to designate the process of deciding *when* and *how*; i.e., which resources to use to execute various activities and over what time frames.

In this paper, we outline a series of models for integrating stand-alone planning and scheduling processes, and summarize the merits of each from the standpoint of agile enterprise control (Section 2). Each model progressively increases the level of information flow between component processes and hence the degree of coupling between them. In Section 3, we consider issues that arise when planning and scheduling must operate over extended periods of time. Such continuous operation requires the ability to perform incremental extensions to plans and schedules, as well as to instigate repairs in response to good or bad partial execution results and unexpected events.

While there have been previous attempts to build integrated planning and scheduling technologies for complex enterprise control (as described in Section 4), the question of how best to design such systems remains open. In Section 5, we briefly describe a research effort focused on developing an integrated planner/scheduler framework in which to explore these issues further. The integrated system will build

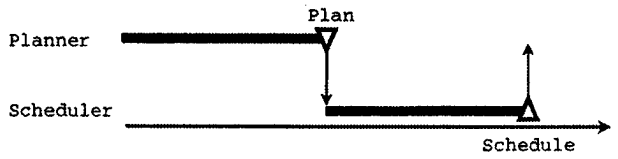


Figure 1: Waterfall Model for Integrated Planning and Scheduling

on component planning [8] and scheduling [14] technologies developed previously by the authors. Both technologies were designed to support incremental change capabilities, and hence are well-suited individually to the requirements of agile enterprise control.

2 Integration Models

The integration of planning and scheduling can proceed along a spectrum from *loose* to *tight* coupling, where coupling can be characterized both in terms of the frequency of interaction and the amount and type of information communicated or shared between them. In this section, we consider several alternative integration models, in order of increased degree of coupling. For each, we describe the basic integration model, requirements imposed by the model on planning and scheduling technologies, and the advantages and disadvantages of each.

2.1 Iterative Waterfall

The simplest approach to integrating planning and scheduling technologies is through an *iterative waterfall* model, very much akin to the classical organizational separation of planning and scheduling mentioned above. Under this scheme, the planner and scheduler operate in sequential, lock-step fashion (see Figure 1). The planner produces a complete set of tasks for achieving a set of objectives (i.e., a complete plan) and then passes them to the scheduler for time and resource assignments. Of course, not all generated plans will be schedulable. Depending on resource availability, some tasks may not be executable within satisfactory time frames or likewise some tasks may need to be dropped from the plan to achieve time and resource feasibility. In such cases, it is necessary to reinvoke the planner to produce an alternative set of tasks. In, general this process will repeat until an acceptable solution (plan/schedule) is obtained.

Realization of an integrated planning and scheduling system based on the iterative waterfall model requires little beyond standard capabilities found in current planning and scheduling systems. Other than the basic plan and schedule generation capabilities, the only requirement is that the planner be able to generate alternative solutions. Ideally, the planner would produce plans that are *qualitatively distinct* with respect to scheduling requirements [9]; otherwise, the differences between successive solutions may be insufficient to eliminate the sources of resource and/or time infeasibility.

In providing a loosely coupled form of integration, the simple waterfall model promotes compartmentalized reasoning. This approach can lead to inefficient and ineffective performance in domains where tasks are resource constrained. There are two contributing factors:

- *Low frequency of interaction* - In the basic case, complete plans are generated before any consideration is given to resource availability concerns, and hence significant effort can be spent expanding strategic process alternatives that could be quickly dismissed due to shortages in required resources.
- *Low information exchange* - Compounding the problem caused by infrequent interaction, there is also a minimal level of information exchange and feedback between planner and scheduler within the iterative waterfall model. In the simplest case, there is no "upward" flow of information from scheduler to planner in cases where a resource-feasible (or otherwise acceptable relaxed) schedule cannot be found, and hence no guidance is provided for producing a set of tasks that is "easier" to schedule. Likewise, no information is passed from planner to scheduler regarding the flexibility (or lack thereof) associated with satisfying various constraints associated with planned tasks. Without such knowledge of input constraints, any relaxation decisions that the scheduler must make to achieve resource feasibility (e.g., slipping deadlines of various tasks) are similarly unfocused from the standpoint of planner acceptability.

2.2 Feasibility Checking

Feasibility checking constitutes one method for increasing the likelihood that generated plans are viable with respect to available resources can be increased. Feasibility checking involves the use of scheduling actions to filter the strategic decisions under consideration by the planner, based on characterizations of the resource requirements that those decisions entail. In short, the scheduler is invoked at intermediate decision points during plan generation to provide estimates of resource feasibility of different planning alternatives.

In standard hierarchical planning approaches [3, 16], strategic choices (which encapsulate high-level tasks or processes) are refined into increasingly more detailed task networks as the planner's search proceeds. Incorporation of feasibility checking into this sort of planning process requires a number of extensions. The *operators* used to encode planning strategies at different levels of abstraction must be extended to include explicit estimates of resource requirements. In addition, it must be possible to configure the scheduler to reason at comparable levels of abstraction. Finally, the planner must be capable of utilizing resource feasibility results produced by the scheduler to discriminate among decision options. In general, the problem of generating effective abstractions of detailed resource requirements

is quite difficult. However, even conservative approximations can provide a basis for significant early pruning of alternatives in specific circumstances.

Feasibility checking can be performed at different levels of granularity and precision. A *capacity analysis* profiles the demand for various resources over time, identifying periods of definite, possible or likely resource contention or oversubscription. Capacity analysis reasons at the level of resource aggregates, rather than individuals, and may also ignore other resource allocation constraints. In cases where relaxed versions of the scheduling problem are solved (i.e., selected constraints are omitted), any detected conflicts are guaranteed to be conflicts in the full problem; however, there is no guarantee that all conflicts have been found. Similarly, it is possible to efficiently compute periods where some possibility of resource contention exists. Depending on the nature of the problem constraints and their interactions, however, neither relaxed formulations nor computation of possible conflicts may provide sufficient basis for conflict detection. In such cases, approximate computation of resource demand profiles may provide a more effective approach. However, this form of capacity analysis introduces the possibility of detecting false resource conflicts.

Capacity analysis provides a quick but possibly incomplete or inaccurate estimation of the resource feasibility of a given plan (or plan fragment). *Resource allocation*, alternatively, provides a more detailed approach by assigning individual assets to specific activities, thus precluding their use by other activities within designated time windows. While resource allocation provides a more accurate assessment of resource feasibility, it is more costly to perform than capacity analysis. The choice of which strategy to use for feasibility checking will depend on the characteristics of the underlying domain.

The key control decision when adopting feasibility checking is the frequency of interaction. Tighter integration (i.e., more frequent feasibility checking) will lead to earlier recognition of infeasible planning decisions, but as the frequency of feasibility checking is increased, both the computational cost and the likelihood of redundant (i.e., non-informative) results also increases. The computational tradeoff can be improved if capacity analysis and resource allocation processes are *incremental* in that they reuse partial results from earlier invocations rather than starting over from scratch for each feasibility check.

2.3 Resource Apportionment

Complex planning tasks usually involve multiple objectives that impose conflicting demands on resource usage. In general, users will have preferences for how resources should be allocated among the tasks selected to achieve those objectives. These preferences might reflect the relative importance or priority placed on different objectives or tasks, as well as their expected consumption requirements. The term *resource apportionment* is used to describe this type of high-level partitioning of resources among tasks. Resource apportionment need not completely assign resources, but rather may simply impose constraints on the resources that can be used by various tasks. Resources might be apportioned at different levels: splitting resource pools among high-level objectives, designating particular resource types for specific tasks, or distributing resources among designated sets of tasks.

Resource apportionment can be seen as a strategy for budgeting sets of resources or resource levels to various portions of a plan. In doing so, resource interactions in the plan are localized, making it possible to consider independently the resource feasibility of various plan fragments. Resource apportionment can be incorporated in a straightforward manner into the integration models discussed above. It can be used, for example, to decompose and decouple the problem of checking the feasibility of various plan fragments associated with different high-level objectives.

Overall, the use of resource apportionment simplifies the resource allocation problem. However, given that it makes certain commitments for resource usage, it limits the range of obtainable solutions and may eliminate otherwise valid (and perhaps superior) solutions.

2.4 Scheduler-Driven Plan Modifications

Resource feasibility checking enables increased efficiency of plan generation through earlier detection of resource conflicts. The *a priori* apportioning of resources provides a complementary benefit, by localizing resource interactions in the developing plan and compartmentalizing resource allocation concerns. While improvements over the waterfall model, neither method exploits information gained during the scheduling process. Rather, the inability to schedule a given plan triggers a restart of the planning and scheduling process, with some internal mechanism of the planner responsible for appropriately redirecting the plan generation process.

Here, we consider models of interaction that are driven by the resource allocation concerns of the scheduler. Within these models, information about resource availability and sources of resource infeasibility is used to redirect the planning process. Although not a prerequisite, this form of interaction is greatly enhanced by an incremental replanning capability. In this circumstance, scheduler/planner interaction can proceed by identifying problematic or sub-optimal aspects of a plan from a resource allocation perspective and suggesting localized modifications to the plan that eliminate the source of these problems.

Problematic aspects of plans can be characterized in two ways.

- *Task-oriented* characterizations identify a set of tasks that cannot be scheduled with respect to current time and capacity constraints. When notified of such tasks by the scheduler, the planner would perform plan revisions to replace them in the current plan with alternatives. Information about additional (perhaps higher priority) tasks that are competing for the same resources could also be communicated by the scheduler, as a way to provide the planner with additional revision guidance.
- *Resource-oriented* characterizations identify those re-

sources/intervals that are oversubscribed. Given a list of such resources, the planner could seek to develop plans that reduce their usage over intervals where they are currently oversubscribed.

Scheduler-driven plan modifications of this type could also be used to support a form of plan/schedule optimization, with the scheduler requesting plan modifications that would improve the overall effectiveness of resource usage. For example, information about excess (or underutilized) resource capacity could provide useful guidance to the plan revision process (as a basis for prioritizing task replacement choices).

3 Continuous Planning and Scheduling

For most complex enterprises, planning and scheduling must proceed on a continuous, open-ended basis. New goals and requests require extensions to existing plans and schedules, while unexpected events may require repairs to previously planned and scheduled tasks, possibly including partially executed activities. Once execution has commenced, time for decision-making is often limited, making it essential that plan and schedule modifications are performed efficiently. Furthermore, *stability* of the plan/schedule is critical, thus requiring that modifications minimize changes. For these reasons, incremental methods for extending and repairing plans and schedules are essential.

Changes to high-level objectives and guidance will require modifications to the plan, which in turn will generally require modifications to the schedule produced for that plan. The planner should succinctly characterize plan changes for the scheduler. Thus, rather than having the scheduler restart from scratch on the modified plan, it should modify the schedule incrementally based on the change in task requirements derived from the plan revisions [13, 17].

Changes to resource availability (either capacity increases or decreases) may necessitate schedule revisions or suggest opportunities for more effective resource usage. Certain resource changes can be accommodated by revisions to the schedule alone. However, more significant changes in resource availability could bring the viability of previously planned tasks into question. In such situations, the planner will need to modify the plan, with the scheduler in turn making corresponding changes to accommodate the revised plan fragment. The decision of whether (and when) to abandon a search within scheduler space for alternative resource assignments and instead request plan changes constitutes an important control problem for planner/scheduler integration.

Unexpected events in the operating environment (e.g., the onset of bad weather) can also impact the viability of both plans and schedules. In general, planners and schedulers incorporate assumptions about both the initial world state, and how the world will change (both naturally, and as a result of executed activities). In the event that such assumptions are violated, plan and schedule revision may be necessary.

To ensure stability, plan and schedule revision processes must be sensitive to the impact that they have on overall activity. Ideally, changes should minimize disruption to those portions of the plan and schedule that are currently under execution, both to ensure continuity and to avoid potentially high costs of redirection. Characterizing what it means to be minimally disruptive is challenging within an integrated planner/scheduler because changes that the planner views as insignificant may prove to be highly disruptive for the scheduler (and vice versa). For example, the insertion of an additional task may be straightforward for the planner to accommodate, but could result in significant rearrangement of resource assignments that could otherwise have been avoided.

4 Integrated Planning/Scheduling Systems

4.1 Characteristics of Planning/Scheduling Problems

It is difficult to identify problem domains that do not require some level of integration of planning and scheduling capabilities. At the same time, differing domain and problem characteristics will dictate varying degrees of emphasis on each capability, and in many cases, one will tend to dominate the problem-solving process.

Consider, for example, the problem of *manufacturing management*, which involves assigning available resources to produce a set of products that satisfy a set of customer orders. For any given set of products to be produced, planning is required to transform high-level specifications into manufacturing process plans. However, assuming that this set of products is to be produced regularly to fill various customer orders over time, scheduling will be the critical determinant of manufacturing system performance (i.e., how well resources are allocated over time). This problem is *resource-driven*, in the sense that the principle concern is to optimize the use of resources to satisfy current demands for known products.

In contrast, problems such as travel planning, project management, or air operations planning have a more *goal-driven* flavor, in that they emphasize the development of resource-feasible strategies for satisfying a fixed set of high-level goals. While effective use of resources is important, the driving motivation is to identify and schedule actions that will ensure attainment of stated objectives. Accordingly, in such domains, planning capabilities will tend to play a more dominant role.

4.2 Survey of Planning/Scheduling Systems

There have been relatively few efforts to develop integrated planning/scheduling systems, with those that have been built motivated primarily by resource-driven applications. Here, we describe a set of systems that are representative of approaches pursued to date.

The early Hubble Space Telescope scheduling application of the HSTS system [6] serves as a good example of a resource-driven approach to integrating planning and scheduling. A set of observation requests for the telescope must be accommodated, each of which requires a number of actions for setup, observation, and clean-up. Within HSTS, an abstract scheduling model is used to optimize the sequence of observations to be taken, and a planner is used

to work out the detailed network of activities for taking each picture. A central contribution of HSTS is a common representational framework that supports both planning and scheduling processes.

The DCAPS [10] and ASPEN [11] systems, like HSTS, provide integrated planning and scheduling environments for transforming high-level goals for spacecraft operations into low-level command sequences. A planning component transforms individual goals into sets of activities that are subsequently scheduled. DCAPS and ASPEN both employ an *iterative repair* model, in which repair methods are applied repeatedly to scheduled activities to improve schedule quality. In addition to effecting direct schedule modifications, repairs can also result in the addition or deletion of activities to be scheduled; in this sense, it can be considered to be a form of scheduler-driven plan modification. The application domains for these two systems share the characteristics that (a) the plans are relatively decoupled, and (b) domain constraints have a strong temporal character. As a result, they both focus on efficient resource allocation and scheduling.

The FORBIN system [2] is an early example of an integrated planner/scheduler. FORBIN adopts a more goal-oriented approach built on a model of tightly-coupled feasibility checking. The action representation language for planning within FORBIN supports a rich model of resource requirements and deadlines. For each planning operation, a temporal simulation is run to test schedulability of the plan that would result. Only planning steps that satisfy the schedulability constraints are considered. FORBIN provides one-time plan/schedule generation capabilities only (i.e., it does not support extension or repair of plans and schedules).

The planning and scheduling system described in [5] is built on an iterative waterfall model, but employs a unique form of feedback from the scheduler to the planner. Based on a probabilistic state model, the planner generates *control plans* that are designed to direct behavior of a device to prevent runtime transition to failure states. Plans are generated relative to a specified probability threshold on states, with higher thresholds resulting in consideration of fewer eventualities and hence simpler plans. Actions within these plans have associated hard deadlines that must be satisfied. The scheduler attempts to generate a periodic schedule for action execution that ensures satisfaction of associated deadlines (and hence avoidance of failure states). In the event that a satisfactory schedule cannot be produced, the scheduler provides feedback to the planner in the form of a recommended higher probability threshold, which results in plans that are easier to schedule. Similarly, when schedules are produced in which resources are underutilized, the scheduler suggests a lower probability threshold for the planner to enable the incorporation of additional activities. The planning/scheduling process then repeats for the new threshold.

IP3S, a blackboard-based system, integrates a generative process planner with a finite-capacity production scheduler [12]. The approach is designed to support a custom, make-to-order manufacturing facility, where approximately half of the received orders entail new products for which process plans do not exist. The system exploits feasibility checking during process plan development to obtain visibility into the current shop load and to generate process plans that avoid current resource bottlenecks. The system is also designed to support incremental management of plans/schedules as updated status information is received. For example, a machine failure resulting in unavoidable resource contention can trigger a replanning process, in which alternative machining processes and resources are explored.

5 JFACC Planner/Scheduler

We have recently embarked on a project to develop an integrated planning and scheduling system that supports generation of plans and schedules as well as adaptations in response to changing tasks, resource availability, or world conditions. The integrated system will accept information relating to feedback from execution assessment, evaluate impact on the current plan/schedule, interleave schedule and plan revision actions as necessary to resolve detected problems and exploit opportunities, and identify/prioritize factors for continued monitoring of the revised plan/schedule. This cycle will emphasize flexible, human-in-the-loop repair, allowing incorporation of robust types of user guidance, constraints and control over the scope, magnitude and types of change.

Our work is motivated by the problem of supporting a Joint Forces Air Component Commander (JFACC) in the development and execution of air campaigns. The characteristics of this domain are representative of many complex enterprises: tasks are complex and open-ended, resource limitations impact the strategies that can be employed for achieving tasks, and the operating environment is both dynamic and hostile. Successful air campaign operations require a mixture of strategy development and accompanying deployment of resources, with tight linkage between them.

The scheduling component for the integration effort is based on OZONE [14], a framework (or toolkit) for configuring reactive and mixed-initiative scheduling systems. The planning and execution monitoring capabilities are based on the Continuous Planning and Execution Framework (CPEF) [8].

5.1 OZONE

OZONE is based on a model of scheduling as an ongoing change process, and is designed explicitly to promote minimally (or selectively) disruptive, incremental scheduling capabilities. At its core is a customizable, constraint-based modeling framework and search architecture, based around three principal components [13]:

- constraint propagation - to incrementally update solution constraints and recognize inconsistencies as changes (extensions, additions, external updates) are made to the schedule,
- constraint analysis - to estimate the critical tradeoffs and opportunities for solution revision (or extension) implied by the current state of the schedule, and

- a set of heuristic scheduling procedures for carrying out specific schedule revisions (or extensions), providing differential optimization and/or conflict resolution capabilities.

The OZONE modeling and scheduling framework consolidates the results of application development experiences in a range of complex, dynamic scheduling domains. One recent application effort has used OZONE to produce a scheduling tool for continuous, day-to-day management of airlift and tanker missions at the USAF Air Mobility Command (AMC) [1]. This "Barrel Master" scheduler is being incorporated into Version 2.0 of AMC's Consolidated Air Mobility Planning System (CAMPS), which is scheduled for operational release in February, 2000.

More recently, we have developed a prototype system for dynamic allocation of munitions and aircraft to air campaign operations which will provide one starting point for our investigation into development of an integrated planner/scheduler.

5.2 CPEF

CPEF embodies a philosophy of plans as dynamic, open-ended artifacts that must evolve in response to an ever-changing environment. CPEF provides a range of operations required for continuous plan management, including *plan creation*, *plan execution*, *monitoring*, and *plan repair*. CPEF has been applied successfully to generate, execute, and repair complex plans for air operations within a simulated operating environment.

Plan creation within CPEF is based on a combination of SIPE–2 [16] and the Advisable Planner [7]. SIPE–2 is a generative planner based on the *hierarchical task network* model of planning [3]. The Advisable Planner provides an advice-taking layer on top of SIPE–2 that enables a user to guide and direct the plan generation so that solutions are customized to his or her individual preferences. Advice enables users to express preferences for strategies with certain characteristics, or that use or avoid specified entities (such as certain resources) in designated situations.

CPEF provides timely adaptation of planned activities based on monitoring of critical events within its operating environment. Conditions to monitor are extracted automatically through a causal analysis of generated plans. A centralized process manager determines when to perform modifications to plans based on monitored conditions and execution results. Plan repair operations are guaranteed to mimize changes to the original plan, and are grounded in the analysis of plan dependency structures [15, 4]. The system also supports *advice-based repair*, in which changes in advice lead to minimally disruptive modifications to the plan that reflect the advice changes.

5.3 Research Objectives

Our goal is to produce an integrated planner/scheduler that is highly responsive to the dynamics of the operating environment. This framework will support a range of core interactions between the planner and scheduler, including resource feasibility testing, resource apportionment, scheduler-based plan revisions, and execution-time plan and schedule repair.

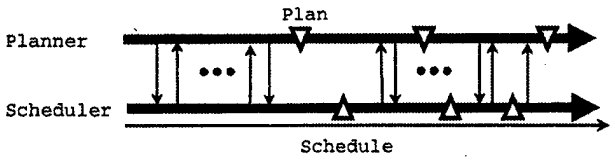


Figure 2: Mixed-initiative Model for Integrated Planning and Scheduling

Our work will involve investigation and development of several novel strategies for implementing those interactions. One such strategy involves the use of the advice-taking capabilities of the CPEF planner to effect schedule-based plan revisions. With this approach, the analysis of scheduling problems will produce a set of *repair advice* designed to eliminate the source of the scheduling problems from the current plan. For example, given an air campaign plan that is unschedulable due to a shortfall in available F-117s for neutralizing enemy air defences within a given sector, advice such as the following would be produced by the scheduler.

> *Avoid unnecessary use of F-117s for attacks on enemy air defences within Sector1.*

The CPEF planner would modify those portions of the plan that relate to the content of the repair advice produced by the scheduler. In this particular case, the planner would reconsider all decisions that resulted in the use of F-117s for the designated attacks, selecting alternative strategies to the extent possible to reduce reliance on the overconstrained resource.

In addition to building novel interaction methods between the planner and scheduler, we are interested in developing rich mixed-initiative models of control for the planner and scheduler that would enable each to make requests of the other to modify their problem-solving strategies, change their current solutions, or provide information about the space of possible changes (see Figure 2). We intend to use the resultant system as a framework in which to conduct experiments for evaluating different models and control strategies for planner/scheduler integration.

6 Summary

The ability to integrate planning and scheduling is critical for enterprise control in dynamic environments, where unpredictable changes necessitate rapid adaptations to both problem-solving strategies and resource assignments. This paper has outlined a series of models for the dynamic integration of planning and scheduling and provided a discussion of their relative strengths and weakness. Previous work on integrating planning and scheduling technologies was

discussed and situated relative to these models. The paper also briefly described a project that seeks to provide tightly coupled planning and scheduling for a highly dynamic and continuous operating environment.

Acknowledgments

The work reported in this paper was supported by DARPA Contracts F30602-97-2-0066 (CMU) and F30602-97-C-0067 (SRI) under the supervision of Air Force Research Lab – Rome.

References

[1] M. Becker and S. F. Smith. Mixed-initiative resource management: The AMC barrel allocator. Technical report, Carnegie Mellon University Robotics Institute, 1999.

[2] T. Dean, J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel time, and resources. *Computational Intelligence*, 4:381–398, 1988.

[3] K. Erol, J. Hendler, and D. S. Nau. Semantics for hierarchical task-network planning. Technical Report CS-TR-3239, Computer Science Department, University of Maryland, 1994.

[4] S. Kambhampati and J. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence*, 55(2):192–258, 1992.

[5] C. B. McVey, E. M. Atkins, E. H. Durfee, and K. G. Shin. Development of iterative real-time scheduler to planner feedback. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1997.

[6] N. Muscettola, S. F. Smith, A. Cesta, and D. D'Aloisi. Coordinating space telescope operations in an integrated planning and scheduling framework. *IEEE Control Systems*, 12(1), 1992.

[7] K. L. Myers. Strategic advice for hierarchical planners. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, pages 112–123. Morgan Kaufmann Publishers, 1996.

[8] K. L. Myers. Towards a framework for continuous planning and execution. In *Proceedings of the AAAI 1998 Fall Symposium on Distributed Continual Planning*, Menlo Park, CA, 1998. AAAI Press.

[9] K. L. Myers and T. J. Lee. Generating qualitatively different plans through metatheoretic biases. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. AAAI Press, 1999.

[10] G. Rabideau, S. Chien, T. Mann, C. Eggemeyer, J. Willis, S. Siewert, and P. Stone. Interactive, repair-based planning and scheduling for shuttle payload operations. In *Proceedings of the IEEE Aerospace Conference*, 1997.

[11] G. Rabideau, R. Knight, S. Chien, A. Fukumaga, and A. Govindjee. Iterative repair planning for spacecraft operations using the ASPEN system. In *Proceedings of the International Symposium on Artificial Intelligence Robotics and Automation in Space (ISAIRAS)*, 1999.

[12] N. M. Sadeh, D. W. Hildum, T. Laliberty, S. Smith, J. McAnulty, and D. Kjenstad. An integrated process-planning/production-scheduling shell for agile manufacturing. In *Proceedings Fifth National Agility Conference*, Boston, MA, 1996.

[13] S. F. Smith. Reactive scheduling systems. In D. Brown and W. Scherer, editors, *Intelligent Scheduling Systems*. Kluwer Press, 1995.

[14] S. F. Smith, O. Lassila, and M. Becker. Configurable, mixed-initiative systems for planning and scheduling. In A. Tate, editor, *Advanced Planning Technology*. AAAI Press, Menlo Park, CA, 1996.

[15] D. E. Wilkins. Recovering from execution errors in SIPE. *Computational Intelligence*, 1(1):33–45, 1985.

[16] D. E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, 1988.

[17] M. Zweben, B. Daun, E. Davis, and M. Deale. Scheduling and rescheduling with iterative repair. In M. Fox and M. Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

A Scalable System Architecture for Autonomous Decentralized Control of Large Adaptive Enterprises*

Alvin S. Lim

Computer Sciences and Engineering
107 Dunstan Hall
Auburn University
Auburn, AL 36849
lim@eng.auburn.edu
phone: (334) 844-6326
fax: (334) 844-6329

Summary

Within a large-scale enterprise system, individual components must be able to respond to internal and external changes autonomously. Development and runtime control environments must allow these internal and external changes to be easilly made in parts of the enterprise and propagated to other parts of the enterprise that will react appropriately and autonomously. Individual components are self-aware and self-reconfigurable which enable them to autonomously participate in adaptive control of large-scale enterprises. The environments provide the appropriate framework for modeling, automatic analysis, scheduling and control of component adaptation in the complex enterprise. Consistent and correct adaptation control requires analysis of dependency and global interactive consistency between the behavior of groups of components in an enterprise.

1 Introduction

Enterprise systems consist of large number of distributed components that may frequently be subject to change from internal and external events, dynamic reconfiguration, incremental development, restructuring and mobility. These components may be either *reactive*, i.e. they continuously react to external and internal events, or *proactive*, i.e. they exhibit autonomous behavior and may generate events independently (self-initiating). Many of these components execute for a long time and may interact in complex ways. Examples of these applications are computer-aided automated manufacturing [4], military command, control and intelligence, and network service applications. Controlling autonomous adaptive enterprises still pose many challenging problems.

Development and runtime control of enterprise systems involve coordination of many distributed components to perform specific tasks. These software controlled systems are modified frequently to adapt to changing information, events, demands and objectives. Components within enterprise autonomously react to these changes appropriately and efficiently. Current techniques in various enterprise environments provide poor support for adaptation, dissemination of change information and proper reaction to changes.

The key to efficient runtime control of large enterprise systems is to build enterprises with smart components that are self-aware and capable of autonomously reconfiguring itself to react to changes in the enterprise. With decentralized control, these reconfigurable smart components allow changes to be made efficiently and incrementally in different parts of the enterprise. They permit global consistency to be maintained among smart components that require correct synchronization and detect liveness problems. Automatic adaptive tools are provided for detecting and eliminating synchronization problems in dynamically changing enterprises.

2 Large Composable Enterprises

There are many examples of large-scale complex enterprises that requires frequent runtime adaptation. These include automated manufacturing, military command and control, business information management systems, mobile inventory systems and

*This research is in part by the National Science Foundation under grant CCR-9896086.

sensor information networks. They can be efficiently developed and controlled using a scalable model that permits composition of multiple levels of control. For example, agile automated manufacturing enterprise may utilize different granularity of control and services [2], such as company business and manufacturing system, factory server, shop controller, workcell controller, workstation controller, automation module controller, equipment controller, device controller and sensor or actuator. A shop controller may control several workcells to ensure that products are produced and dispatched at their actual or expected date as required on demand by the company business and management systems which in turn are responsible for planning and satisfying the rapidly fluctuating demands of the customers. Each workcell may contain several workstations and equipment. A workcell controller manages different sequences and types of operations that may involve synchronization, recovery, and adaptation to different manufacturing task requirements. We need to synchronize the operation of the individual machines and tools for correct collaborative jobs. The behavior of a workcell is encoded in a composite component that controls its operation. In agile manufacturing domain, components that control the machine and tools must be adapted, reconfigured and replaceable without shutting down the workcell. Efficient agile manufacturing enterprise must address the problem of maintaining consistency of other components that interact with the components that are being adapted and reconfigured.

3 Adaptable Components

Large enterprises are built from adaptable components that may be developed independently but may interact with other adaptable components through well-defined interfaces that encapsulate interaction properties. Components execute autonomously in a decentralized environment and may perform and control various operations in the enterprise. They may simultaneously be service providers for other components and clients of services that other components provide. New components may be added into a cluster of components and register their services with a resource server responsible for the cluster (Figure 1). A resource server may contain information on services at multiple clusters. Other components that require a service will discover the services available in a cluster through the resource servers that return the location of the service components, similar to Jini [1]. The client components then interact directly with the server for the service.

When components are moved to new locations or when enterprise adaptation is required, services provided by the components may be removed from a cluster and relocated to another cluster without manual reconfiguration. Components may be mobile and can be easily reconfigured in an enterprise. They are self-aware of their own location, configuration and services that they perform. These components are self-reconfigurable and efficiently support dynamic adaptation and mobility of enterprises. Correctness of adaptation is ensured by the adaptation server.

4 Services for Adaptive Control

Services required for control of components and groups of components within a large enterprise are provided by compositional servers, adaptation servers and resource servers that support analytical tools and adaptation mechanisms (Figure 1). Servers may be replicated for higher availability, efficiency and robustness. Servers may also be distributed which requires coordination to perform the service, e.g. distributed resource servers may work together to discover the location of a particular service requested by the application client.

4.1 Compositional Server

The compositional server manages various components that may be added into (or removed from) clusters in the enterprise. It also manages abstractions (group behavior) of clusters and hierarchical composition of component clusters. The compositional server simplifies dynamic reconfiguration of services provided by each component or cluster. Development of large enterprise is simplified by allowing individual components to be specified and designed independently while the compositional behavior and constraints on a cluster of components may be separately specified.

4.1.1 Enhancing Compositionality and Abstraction

To enhance adaptivity in enterprises, each component is designed independently and the interaction and synchronization operations with other components may be described separately. This decoupling of autonomous components from their synchronization and interaction behavior enables components to be easily adapted, replaced and reconfigured when triggerred by dynamic events in the enterprise. Clusters of components may provide abstract aggregate services performed by coordinating the collaborative tasks among the components. Clients of the services may work cooperatively together to perform common objectives in the abstract services. The formalism used for defining separate component behavior and

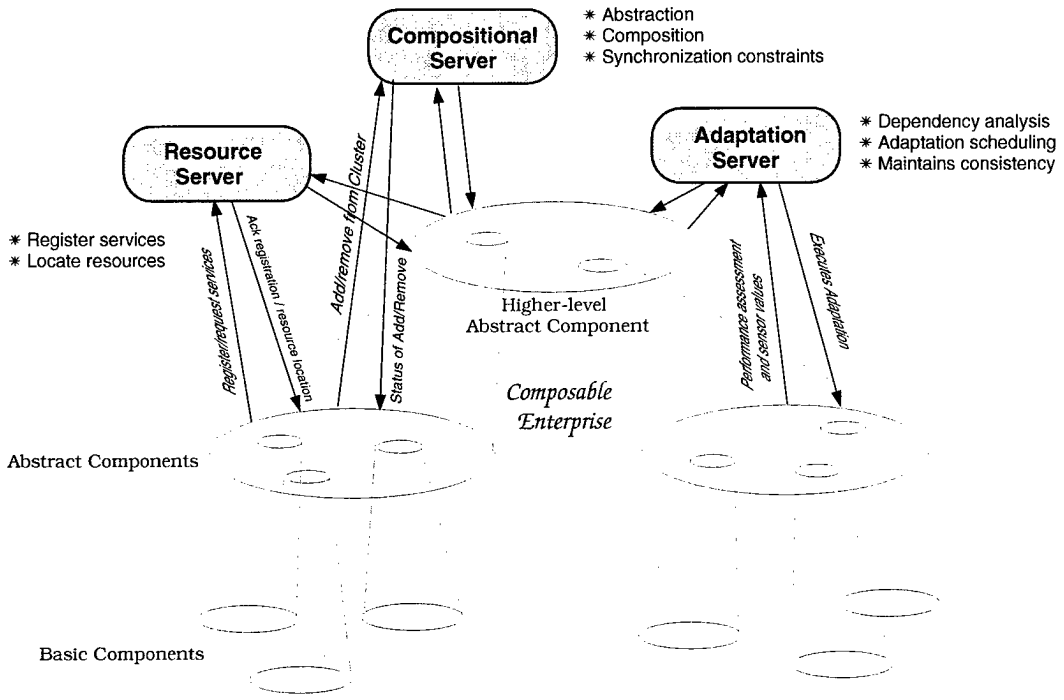


Figure 1: Services for Adaptive Control

interaction behavior simplifies the mechanisms for replaceable components and alterable interaction patterns.

4.1.2 Composite Component Hierarchies

Components may be hierarchically composed. Each composite component may be used to form a higher level composite components. Individual component's local and interaction behavior are hidden from composite component at higher levels. The capability to specify hierarchical composite components enables designers to build large and complex enterprise systems by clustering together smaller components at each level. Hierarchical clusters of large number of autonomous components can be managed in a scalable way by distributed compositional servers. Each composite component defines a limited scope in which the behavior of a cluster of components can be independently analyzed and controlled.

4.2 Adaptation Server

Adaptation servers utilize information from the compositional server to control components during dynamic adaptation and failure recovery of the enterprise. Each component may execute autonomously to control different operations in the enterprise and may interact and coordinate independently with other components to perform collaborative tasks. For example, in a flexible manufacturing cell, a component may control a milling machine and may communicate with components that control the robot and workpiece to coordinate the task of removing a milled workpiece [4] and transporting it to the next workstation.

Adaptation servers monitor component clusters during normal execution through sensors, polling of the components or explicit end-user directives for reconfiguration and failure recovery. When a runtime reconfiguration is requested or triggerred, the adaptation server will generate the appropriate schedule of reconfiguration operations that will ensure the reconfigured and affected components are globally consistent. The adaptation server is then responsible for ensuring the reconfiguration operations are properly executed as scheduled. To improve performance and autonomy, we allow an hierarchical application structure where the compositional and adaptation servers may coordinates with other higher-level and peer servers. Servers are not customized for each application. Instead, they store and analyze the component composition and analytical results for coordination, adaptation and failure recovery.

4.2.1 Analytical Tools

When components are added or removed from the enterprise, analysis may be required to ensure that the enterprise still maintains its safety and liveness properties. Components (or clusters of components) may be specified and analyzed independently of the distributed applications in the enterprise. A generator may take the component and synchronization (interaction) specifications and produce intermediate results to a diagnostic checker that checks for syntactic correctness, overall reachability of all states, and feasibility of each abstract behavior. The suite of analytical tools automatically verifies the potential for deadlock, livelock and starvation. While deadlocks are usually easily removed by specifying additional constraints, livelock and starvation are probabilistic properties which are not easily removed by modifying the specification. Starvation and livelock are better handled by dynamically detecting them and recovering affected processes. The analytical tools also analyze dependencies for preserving global consistency during dynamic adaptation and failure recovery.

4.2.2 Decentralized Control

To enhance scalability, we allow components to be hierarchically composed and controlled by distributed adaptation servers. Each composite component cluster may be used as an abstract component to construct higher level composite component clusters by a higher level compositional server where only abstract composite behaviors are visible to composite component cluster at higher level. Each lower level composite component cluster is controlled independently by a different compositional server. The information on components and analytical results stored in the compositional server may be used by the runtime adaptation server to prevent liveness problems and maintain consistency during dynamic adaptation and failure recovery of parts of the enterprise.

4.3 Resource Server

New services may be provided (or deleted) by a component to other components in a cluster. A component may register a service that it can perform with a resource server. Each component has a home resource server which keeps track of the location of the component when it moves. Other components that may require the service may request the service from a resource server. If the service is recorded in the resource server, it will return the location of that service to the requesting component. Otherwise, a discovery protocol is used to locate the service through other resource servers. At intermittent frequency, service registration information may be disseminated from one resource server to other resource servers in the enterprise. Services that are registered with the resource servers may involve multiple distributed components with complex nonlinear interaction.

When a component move to a different cluster at another location, it notifies the previous resource server that it is moving. When it arrives at another cluster in a new location, it will register with the new resource server which will notify the previous resource manager and the component's home resource manager. Existing interactions between the mobile components and other components will thus be handed over to the new location.

5 Adaptation Planning and Control

Enterprises may adapt to external and internal events by reconfiguring its components and functionalities dynamically. There are two ways in which adaptation may be initiated: (1) Explicit request by the human operator of the application, and (2) trigger from sensor values that exceed certain thresholds.

Adaptation requests may be made (or triggerred) to the adaptation server by specifying (1) the required adaptation operations and (2) the necessary adaptation constraints. Adaptation operations may include methods for modifying services, replacing them or introducing new services. These adaptation may occur concurrently with other normal operations of the enterprise. Adaptation operations for adding, removing or modifying services are performed by calling the compositional servers. During adaptation, new components may be added which may interfere with other components. Adaptation constraints that define interaction and synchronization constraints may need to be specified to ensure correct behavior while adaptation is being carried out.

From the set of adaptation operations, the adaptation server automatically generates a schedule of adaptation operations that preserves global consistency. Adaptation of higher level composite components is performed in similar ways, where adaptation operations are applied to abstract clusters of components. This will involve calls to the compositional servers that maintain the abstract behavior of the clusters. The compositional server controlling the abstract components will propagate the adaptation operations to the lower components abstractions successively at each level.

5.1 Consistent Schedule

Dynamic adaptation of complex enterprises require appropriate planning to ensure the adaptation operations maintain global consistency as specified

by the service requirements. The analysis for correct adaptation schedules may involve clusters of interacting components and higher levels of component clusters with abstract services. By analyzing abstract behavior of clusters, we reduce the size of the problem space used in the analysis, making it scalable to very large enterprise systems. Dynamic adaptation may cause inconsistency in an enterprise. The analyzer determines a correct adaptation schedule from the synchronization constraints and dependency information stored at the compositional server.

From the analysis, components not directly involved in the adaptation operations may still be affected by the adaptation. These affected components may themselves be required to perform adaptive operations to restore consistency in the enterprise after the adaptation is completed. Although recovery and reconfiguration operations may cause inconsistency in the current behavior, consistency may be restored by further recovery in the new configuration. Formal methods may be used to ensure correctness of the dynamic adaptation. These policies may be supplied by the designer and stored in the compositional server. The framework provides mechanisms that can be applied to any application-specific policies for enforcing correct adaptation of enterprses.

5.2 Runtime Adaptation Control

The adaptation analyzer provides the runtime adaptation server with the correct schedules of adaptation operations for the affected components. The adaptation server manages the appropriate adaptation whereby new components may be added, removed, replaced, modified or reconfigured. Some of the these adaptation operations are performed through the compositional server. During the adaptation, synchronization constraints are satisfied, but components may be temporarily inconsistent. However, consistency will be restored at the end of the adaptation. The mechanisms allow different policies to be used for enforcing consistency in the enterprise. The adaptation server controls synchronization and adaptation by receiving adaptation requests from the components and initiating adaptation operations in the affected components according to the adaptation schedules.

The adaptation server will use information from the compositional servers and resource servers to ensure that adaptation operations will not lead to synchronization problems or eventual inconsistency in the affected components. After adaptation is completed, the compositional servers will update its information on the new cluster composition and behavior. Resource servers will also be updated to include the new services provided by the new components and clusters. New service information are also disseminated to other resource servers in clusters that may require the new services.

6 Adapting Large Dynamic Enterprises

Large dynamic enterprises, such as flexible manufacturing and military command and control, involve dynamically changing structures and control. The compositional servers, resource servers and adaptation servers work together to facilitate dynamic changes in the enterprises and maintain crucial compositional information for interacting components in the enterprise. Components may be added, removed and moved around the enterprise using the compositional servers that maintain the enterprise structural and interaction information. Components may publicize the services they provide by registering with the resource servers. Other components that need a particular services may request for them from the resource servers. Services from mobile components may be relocated to resource servers nearest their current location. Resource servers are responsible for dynamic handover of active services interaction. The adaptation servers are responsible for the correctness and global consistency of the applications when the enterprise is engaged in either spontaneous (triggerred by existing conditions or events) or planned dynamic changes in its structure and control.

In the agile automated manufacturing enterprises, the behavior of a workcell is controlled by its composite controller component that is hierarchically composed of components that controls the workstations, machine tools, sensors and actuators used in the workcell. The composite component that manages the enterprise at the shop floor level is simplified since it manages workcell components rather than components at the sensors and workstation levels. At any time, components may join (or leave) the enterprise and be automatically connected (or disconnected) to the clusters and communication infrastructure through the compositional and resource servers. Once connected, they may interact with other components in the enterprise to perform coordinated tasks.

When the workcell needs to be reconfigured because of a sudden increase in the demand for a certain product, the workcell can be efficiently adapted on the fly, i.e. without shutting it down, for the new product. The previous operations in the workcell to produce the old product may continue to completion while parts of the workcell are being reconfigured for the new product. The adaptation request for a set of adaptation operations to be performed on the work-

cell may first be sent to the adaptation server which analyzes the dependencies between operations currently being performed in the workcell. From the dependency analysis, the adaptation server determines which workstations or machine tools will be affected by the adaptation operations. While the required adaptation operations are be executed, the affected components may require additional adaptation operations in order to maintain global consistency eventually.

7 Related Work

We are interested in a common scalable system architecture where different formalisms may be utilized to enforce correct adaptation and synchronization of the components in enterprises. Other adaptive environments, including Darwin [7], ILI [8], and Polylith [10], while allowing structural adaptation, do not enforce consistency and correctness policies for adaptation. These responsibilities are left to the application designers. Conic [6] and Argus [3] use transactions for preserving consistency, but restrict complex interaction and operations of enterprises. Formal methods are useful for specification and analyses of large complex enterprises and may be utilized in this framework. Formal methods that are applicable here are Statecharts [5] and hybrid automata [9]. Hybrid automata is a useful model for real-time hybrid systems (e.g. physical plant control) that contains two distinct types of systems – continuous and discrete-state – that interacts with each other.

This scalable framework allows new adaptation techniques based on transactions, although some may be restrictive, particularly for enterprises with complex interactions. Newer extended transactions, mobile transactions and non-transactional mechanisms that do not have these restrictions may also be supported for enterprises that requires those flexible capabilities. During normal operations, these techniques may not require serializability and permits complex non-commutable interactions. During adaptation of the enterprise, components will be automatically restored to global consistent states using analyses of dependency constraints.

8 Conclusions

We presented a scalable system architecture for decentralized control of enterprises that are composed of autonomous components and may be reconfigured dynamically to adapt to internal and external events. It supports hierarchical composition of components in which incrementally higher levels component clusters may provide abstract services. This support for compositionality and abstraction enhance adaptability in large enterprises. Using formalisms that separate the localized operations of components from their interactions with other components, the analytical tools may be utilized to ensure correct and consistent adaptation at runtime.

References

[1] Arnold, K., et. al., "The Jini Specification," Addison Wesley, 1999.

[2] Biemans, F., et. al., "Computational Tasks in Robotics and Factory Automation," *Computer in Industry*, 10, 1988, pp. 95-112.

[3] Bloom, T., Dynamic Module Replacement in a Distributed System," TR MIT/LCS/TR-303, *MIT Lab. for Computer Science*, March 1983.

[4] Duffie, N., et. al., "Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities," *Journal of Manufacturing Systems*, V. 7, N. 4, 88, pp.315-328.

[5] Harel, David, "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming*, August 1987, pp. 231-274.

[6] Kramer, Jeff and Jeff Magee, "The Evolving Philosophers Problem: Dynamic Change Management," *IEEE Transactions on Software Engineering*, Vol. 16, No. 11, November 1990, pp. 1293-1306.

[7] Magee, J., A. Tseng, and J. Kramer, "Composing Distributed Objects in Corba," *International Symposium on Autonmous Decentralized Systems*, 1997.

[8] Martin, V. and K. Schwan, "ILI: An Adaptive Infrastructure for Dynamic Interactive Distributed Applications," *Int. Conf. on Configurable Distributed Systems*, May, 1998.

[9] Nerode, A. and W. Kohn, "Models for Hybrid Systems: Automata, Topologies, Controllability, Observability," *Workshop on Theory of Hybrid Systems*, LNCS 600, pp. 317-356, 1991.

[10] Purtilo, J., "The Polylith Software Bus," *ACM TOPLAS*, January 1994.

Real-time Symbolic Control of a C^2 Enterprise

Azad M. Madni, Ph.D.
Intelligent Systems Technology, Inc.
2800 28th Street, Suite 306
Santa Monica, CA 90405
310-581-5440, Fax: 310-581-5430
amadni@intelsystech.com

Abstract

Real-time enterprise management and control is a continuing challenge facing the C^2 community. Much of the R&D to date has centered around AI-based techniques for planning and scheduling with limited work on intelligent process management that integrates planning and execution. Today there has been a surge of interest in control-theoretic approaches that offer formal methods and metrics to evaluate system properties such as controllability and stability – characteristics which are crucial for mission-critical applications. Along with this interest has come the challenge of extending control-theoretic concepts to the domain of symbolic control.

This paper presents a symbolic control formulation of military C^2 operations. This formulation adapts and combines key concepts and technologies from control theory, decision theory, agent-based systems, dynamic programming and fast-time simulation. The paper identifies potential sources of instability and presents an experimentation framework for defining and investigating key properties such as controllability and stability of a symbolic control system. Two illustrative examples are presented. The first illustrates a representative set of transformations that are involved in symbolic control and suggests the use of complementary techniques from multiple disciplines in a way that exploits their respective strengths while overcoming their limitations. The second shows the mapping from a symbolic to a numeric domain for higher levels in the C^2 enterprise. Specific metric for evaluating mission success are provided for each example.

1. The Real-Time Enterprise Process Control Problem

Real-time enterprise management and control is a continuing challenge facing the C^2 community. Much of the R&D to date has centered around AI-based techniques for planning and scheduling with limited work on intelligent process management that integrates planning and execution [1, 2]. Today there has been a surge of interest in control-theoretic approaches that offer formal methods and metrics to evaluate system properties such as controllability and stability – characteristics which are crucial for mission-critical applications. Along with this interest has come the challenge of extending control-theoretic concepts to the domain of symbolic control.

The essence of the problem in achieving closed-loop, agile, stable control of military operations is contained in the following paragraph:

There is no way of determining whether or not the defined set of activities/actions in a plan can achieve the desired end state from the initial state or any intermediate state. What is needed is an approach for predictable, stable and agile control that can ensure that the desired end state is reachable, and the operations are agile, flexible, stable, and robust.

In the C^2 context, (a) *agility* is defined as the ability to rapidly and cost-effectively adapt to dynamic change in the battlespace, e.g., guidance, resources, situation; (b) *flexibility* is defined as the ability to support different conflict scenarios (e.g., theater battle, brushfire warfare, counter-insurgencies, urban versus rural conflict, peace-keeping) phases and campaign phases; (c) *stability* and *robustness* are defined as the ability to keep decision-making processes synchronized with the pace of operations and provide predictable, bounded outputs from the decision-making processes in response to perturbations in inputs, plant parameters, and the external environment. This paper presents a symbolic control formulation for management and control of a C^2 enterprise and defines key concepts such as controllability and stability within the symbolic control formulation.

2. Control-Theoretic Formulation of a C^2 Enterprise

The Observe-Orient-Decide-Act (OODA) loop, defined by John Boyd [3] captures the iterative nature of processes within a military warfighting enterprise. It recognizes that the results of our actions, and the opponent's subsequent actions, or at least our observation of those actions, become part of the next input. The mapping of the OODA loop within a control system construct is shown in Figure 1. As shown in this figure, feedback information consists of a fresh set of observations with which to re-orient.

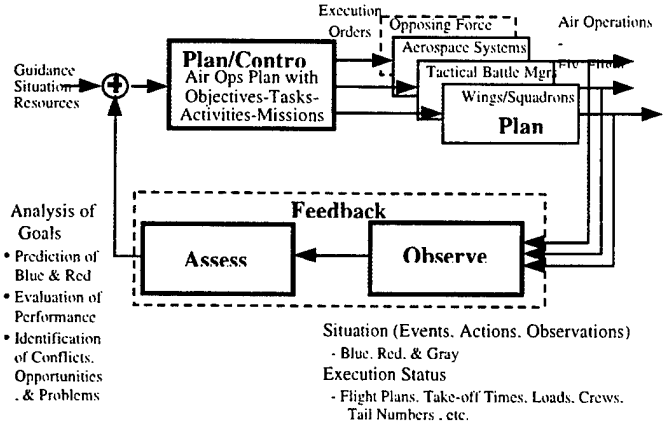


Figure 1. Adaptation of the OODA Loop for C^2 Enterprise Control (Source: JFACC Broad Area Announcement)

The major blocks in Figure 1 can be interpreted within a C^2 problem domain. In this interpretation, the controller Block is the Command Center of the Blue Force, the "plant" consists of Blue Battle Assets, and the Observe/ Assess Block provides feedback on the tactical situation and state/status of the plant as well as the Opposing Force.

As shown in Figure 1 the control consists of symbolic commands (e.g., from a Command Center) to the plant (e.g., own battle assets) which are intended to create/ influence the behavior of the plant and achieve the desired effect on the environment (i.e., the Opposing Force). The results, which are in the form of state and status of the various subsystems of the plant, are observed and assessed. The execution and direct observation of actions may or may not be symbolic. The assessment, which is symbolic, is reported back to the Command Center. Based on these results, the Command Center issues the next command.

3. Symbolic Control (SC)

Many realworld systems are controlled by qualitative or symbolic commands. Symbolic control refers to the use of high-level symbolic commands to control a "plant." The feedback from the plant can be a combination of numeric and symbolic information. Symbolic control has seen use in applications such as robotic manipulator control, robot movement control, manufacturing assembly lines, and to a very limited extent, in C^2. Examples of symbolic command are "shut valve," "turn left," "retreat," "attack." An example of symbolic feedback is "sensor #2 inoperative."

Prior research in symbolic control has included a system that was controlled using feedback consisting of symbolic information [4] and C^2 reference models that were driven by symbolic commands [5], [6]. Common to these research prototypes was a high-level decision-making body such as a human commander that made control decisions with the help of symbolic variables.

4. Challenges in SC of C^2 Enterprises

Historically, symbolic control has been successfully applied in domains, where there exists precise knowledge of various components of the plant. For example, in robotic control, there exists a precise function to map a command to its observable result (i.e., outcome). Specifically, the result of a command can be observed and that knowledge used to decide the next command. A decision such as this can be made with a degree of confidence because the results or outcomes of all possible commands are known a priori. However, in a complex system, it is not possible to predict the effects of interactions of the behaviors of its various components. A C^2 enterprise is a complex system.

In a C^2 enterprise, the observation and assessment are the result of ALL the previous commands and other "forces" at work (i.e., it is cumulative). For certain commands, it may be possible to have an accurate prediction about their results based on previous execution of similar commands (i.e., prior cases). However, for certain other commands, this simply may not be possible. For example, in some cases we may not be able to account for all the "forces" that could affect an outcome (i.e., result). Similarly, when we observe an outcome, we may not be able to isolate the command responsible for that outcome. This is a fundamental issue in a closed-loop control formulation. Without a satisfactory answer to this problem, we can end up with what appears to be a properly operating controller that, in fact, is issuing erroneous commands because it is operating with and on incomplete or incorrect information! This condition will inevitably lead to various types of instability.

The typical operational challenges in adaptive control pertain to controllability, observability, stability, agility, performance, time delays, and robustness of the system. For C^2 enterprises, these problems are further exacerbated in that commands are symbolic, feedback can be symbolic or numeric, and the plant has to be characterized by non-algorithmic methods (e.g., qualitative models or state machines) at higher levels in the enterprise. The implications of the former are that we need to be able to: (a) transform commands from the symbolic domain to the numeric domain, perform processing in the numeric domain and then map the results in consistent and complete fashion to the symbolic domain. The implications of the latter is that "plant" representations should be rich enough in terms of observables to support the assessment function.

5. Potential Sources of Instability

With respect to Figure 1, there are several general conditions that can lead to potential instabilities. Table 1 shows the problems that can arise, their potential implication on the control system, and the likely impact on mission accomplishment.

Examples of specific sources of instability in a C^2 system are: (1) loss of communication with a support group or platform; (2) opposing force employs jamming or surprise tactics; (3) the model of the plant is an incomplete representation (of plant behaviors) resulting in divergence of the model-predicted behavior and actual plant behavior; (4) battle assets are temporarily unavailable or destroyed; and (5) tactical events occur at a rate that is faster than event processing time resulting in either certain events going unattended, or oscillatory behavior of the closed loop control system.

Table 1. Potential Problems in SC Systems

Control System Component	Potential Problems	Likely Result/ Implication
Observe/ Assess Block (feedback)	• Partial feedback (some plant states unobservable)	controller could issue erroneous command/goal achievement compromised
	• Absence of feedback (unobservable plant state and status)	open-loop control/ likelihood of undetected or uncorrected instabilities
Controller Block	• Erroneous command (loss of feedback, partial feedback)	does not achieve desired impact on plant/goal achievement compromised
Plant	• Temporary shortage of a resource	Potential violation of event servicing time constraint leading to unstable C^2 system operation/can compromise mission success
	• Inadequate resources	cannot achieve desired impact on plant/goal achievement compromised
	• Unexpected plant behavior	no suitable command available/goal achievement compromised
	• Rapid plant state changes/events occurrences	not enough time to process change events/ likelihood of instabilities/ oscillations
Human Decisionmaker	• Erroneous input/over-ride • Inappropriate intervention	unstable system behavior/ can compromise mission

6. SC System Design and Evaluation Framework

To achieve symbolic control of a large-scale system (e.g., C^2 military operations) requires a framework that enables investigation into various control issues such as stability, robustness, controllability, and agility. The guiding design principles are shown in Table 2.

Table 2. Guiding Principles

- Employ an *overarching control-theoretic construct* as the organizing metaphor.
- *Partition the control problem* into manageable, tactically meaningful blocks.
- Develop *consistent transformations* (for a class of problems) that map symbolic control commands to numeric commands (and vice versa).
- *Use appropriate techniques* from different disciplines (e.g., AI planning, optimization, decision theory, simulation) within each block.
- *Model the plant with sufficient fidelity* to enable assessment, MOE evaluation, and experimental validation of key hypotheses.
- *Model the Opposing Force with just enough fidelity* to explore stability and robustness regimes of the controller.
- *Define properties* such as stability, controllability, robustness, and agility and refine these definitions within the domain of interest and in light of the representation metaphor used.
- *Define metrics* that: (a) support evaluation of mission success; and (b) are directly measurable or have measurable "proxies."

Figure 2 shows our symbolic/numeric/symbolic transformation (SNST) study framework. This framework encompasses mapping symbolic commands to numeric commands, performing analysis and manipulation in the numeric domain, and correctly mapping the numeric results back to the symbolic domain

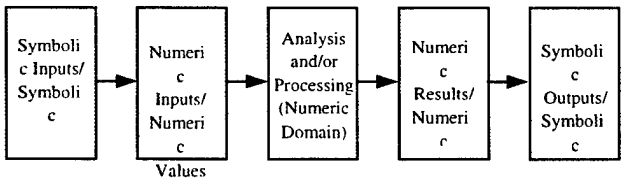


Figure 2. Symbolic/Numeric/Symbolic Transformations (SNST) Approach

As shown in this figure, symbolic commands/inputs from the symbolic feedback system are transformed from the symbolic domain into numerical commands/inputs using appropriate mappings/transformations. The analysis and processing is done in the numerical domain using appropriate algorithms. The numeric results are then mapped back into outputs in the symbolic domain that can be easily understood (both in qualitative as well as quantitative terms) by human decisionmakers. Based on this study framework, the following research hypotheses can be pursued.

Modeling. The SNST approach shown in Figure 2 can be applied to the symbolic feedback block of Figure 1. *The key hypothesis that we want to validate is that appropriate useful transformations and maps can be developed from symbolic to numeric and numeric to symbolic domains for C^2 enterprises.* Of course, issues of consistency and units have to be addressed so that the results following each transformation continue to be meaningful, consistent, and accurate. For example, all possible commands could be put in a symbolic set which could then be transformed into a numeric set by associating each command with a number, or a set of numbers or a set of equations. The numeric set could then be dealt with numerically, and after suitable processing, the numeric results would belong to a certain numeric set which could then be transformed into another symbolic set. The appropriate transformations in the SNST approach can be created by using examples and real data to validate this hypothesis.

The implications of validating this hypothesis for a C^2 warfighting enterprise are as follows. In parallel warfare, the enemy's capability is described as a system of systems (e.g., transportation, C^2, power distribution, communications, forces in the field). The key task in this situation is to select the optimum set of targets spread across all the systems (each of which modifies the capability of that particular system to some degree) that maximizes the impact on overall goals such as causing the enemy to withdraw/retreat.

The systems, target list, weapons available, system functional capability are symbolically represented in the C^2 planning and execution domain. We can hypothesize that these symbols can be mapped into the numerical domain (i.e., numeric values, algebraic or differential equations, expressions) and then manipulated automatically to select the optimum strategy. The numerical solution can then be mapped back (i.e., transformed) into symbols that are understandable, and consequently, executable, evaluatable, and manipulable by humans.

Observability/Controllability. Observability and controllability are fundamental concepts [7] that need to be defined and understood within the framework of symbolic control. For example, if a certain event occurs within the system but the information contained in the observations is inadequate for event detection and assessment, then that could mean that the system (i.e., the way we model it and for the number of observations we are able to collect) is unobservable. Similarly, system states that cannot be affected by any commands we apply could be termed uncontrollable. These concepts have to be made precise in the context of the symbolic world before being related to numerics using the SNST approach. *The key hypotheses here are that observability and controllability concepts can be made precise in the context of symbolic control, and that SNST transformations can be designed to preserve observability and controllability.*

Stability. The stability of the system in Figure 2 can be viewed in two different ways using concepts from Discrete Event Systems [8, 9]. First, stability can be defined as a set of acceptable states of symbolic variables. Thus, when the variable in question is outside the acceptable set, appropriate processing should be done to bring the system back into the acceptable set. For example, in any conflict scenario, own destruction may be an unacceptable state whereas destruction of the enemy without subsequent negative consequences could be a

desirable state or an "equilibrium" state (at lowest energy) as defined in the numeric world.

Second, stability can be viewed as a set of acceptable system behaviors. This is a less restrictive definition in that the system can go through unacceptable states defined above as long as the system continues to exhibit overall desirable behavior. For a warfighting system, this means that there may be temporary setbacks such as sustaining damage to one's assets. But as long as recovery is possible and enemy destruction is achievable, it may still be viewed as an acceptable system behavior. The concept of stability and the various forms of stability can be developed for the symbolic feedback block of Figure 1 using concepts from discrete event and nonlinear systems. The consistency and invariance of these concepts in the SNST approach can also be investigated. *The key hypothesis here is that stability can be defined as a set of acceptable states of symbolic variables. The second hypothesis is that a system can go through unacceptable states temporarily as long as it continues to exhibit desirable overall behavior. The third hypothesis is that stability concepts and their different forms can be developed for the symbolic feedback block in Figure 1 using concepts from discrete event and nonlinear systems.*

Performance. Performance, a crucial metric for any control system [7], is difficult to assess for systems controlled through symbolic (variable) commands. Keeping the final objective in mind, a study has to be conducted to define a performance index that drives the system towards the achievement of the goal. The performance metric has to be specific to the situation and desired goals. The SNST approach could be used to convert symbolic performance measurements into numeric values, and vice versa. A numeric performance metric could be used for analysis and numeric processing as part of the overall SNST approach. *The key hypothesis here is that it is possible to define a performance metric which can be used by a symbolic controller to drive the system toward achievement of the goal.*

Robustness. A control system should be robust to changes in: a) the 'plant' that is being controlled, and b) with respect to external disturbances [7]. In a symbolic world, the situation is quite similar. Changes within the system due to unpredictable decisions or moves of the enemy could be viewed as unknown changes of the model. Unexpected or "unmeasurable" events that affect the response of the system could be viewed as external disturbances. In this case the command and control inputs should be such that the changes in outcome affected by changes in the model and/or external disturbances is minimized (i.e., the outcome is bounded). In the SNST approach, this robustness analysis can be performed in advance by analyzing the characteristics of the numerical processes to create the effects of hypothetical, unspecified events and determine appropriate bounds to prevent unacceptable system responses. Without this strategy, the actual implementation could prove costly with undesirable and/or unpredictable results. *The key hypothesis here is that an intelligent controller can adapt to pertubations in the model parameters and/or external disturbances and produce bounded output(s).*

Time Delay. Time is always a crucial factor in realtime dynamic systems [7, 8, 9]. In the symbolic system (Figure 1), the time interval within which observations of the system are collected, assessed, and processed to generate the next command and control input, is important. Significant time delays can affect performance and often lead to instabilities. With SNST, one could estimate bounds for the allowable delays using the numeric block that guarantees stability and performance. *The key hypothesis is that for an "observe-assess-respond" cycle time that is strictly less than the duration between key external events, the system will be stable (i.e., no oscillations).*

7. Illustrative Example: Air Intercept

This illustrative example is concerned with the air intercept operations of a fighter aircraft. It is meant to illuminate two key points: (1) an appropriate series of transformations can be created between the symbolic and numeric domains and vice versa; and (2) by "mixing and matching" techniques from various disciplines we can exploit their respective strengths while compensating for their respective limitations [10, 11, 12].

The Scenario

The Combat Air Patrol (CAP) role for a fighter aircraft within the Fleet Air Defense Mission consists of twelve phases. The specific phases of interest in our illustrative example are Phases 5, 6, and 7.

Phase 5 is station-keeping/loiter. In this phase, the combat aircraft adheres to a patterned flight at a designated position from the task force. This phase terminates when patterned flight ceases upon target detection.

Phase 6 is target intercept. In this phase, the aircraft pursues a flight path toward a relative position (target conversion) on a selected airborne target. This phase terminates when both the intent to launch a weapon and the capability to effectively launch a weapon exists.

Phase 7 is air-to-air combat. In this phase, the aircraft is flown within a selected weapon launch envelope against a specific target. This phase includes beyond visual range (BVR) and within visual range (WVR) engagements. The air-to-air combat phase terminates when no further launch capabilities exist or are desired, and the desired return altitude and speed profile has been attained.

Symbolic Control Formulation

Figure 3 shows a symbolic control formulation of the air intercept planning and control operations of a fighter aircraft in a Combat Air Patrol (CAP) role. The recommended technique from different disciplines is shown within each block in this figure. The CAP role

starts with target detection and culminates with target re-attack. The overall mission objective can be summarized in three key objectives that act in tradeoff fashion: (1) maximize carrier safety; (2) maximize tactical gains; (3) minimize resource expenditure.

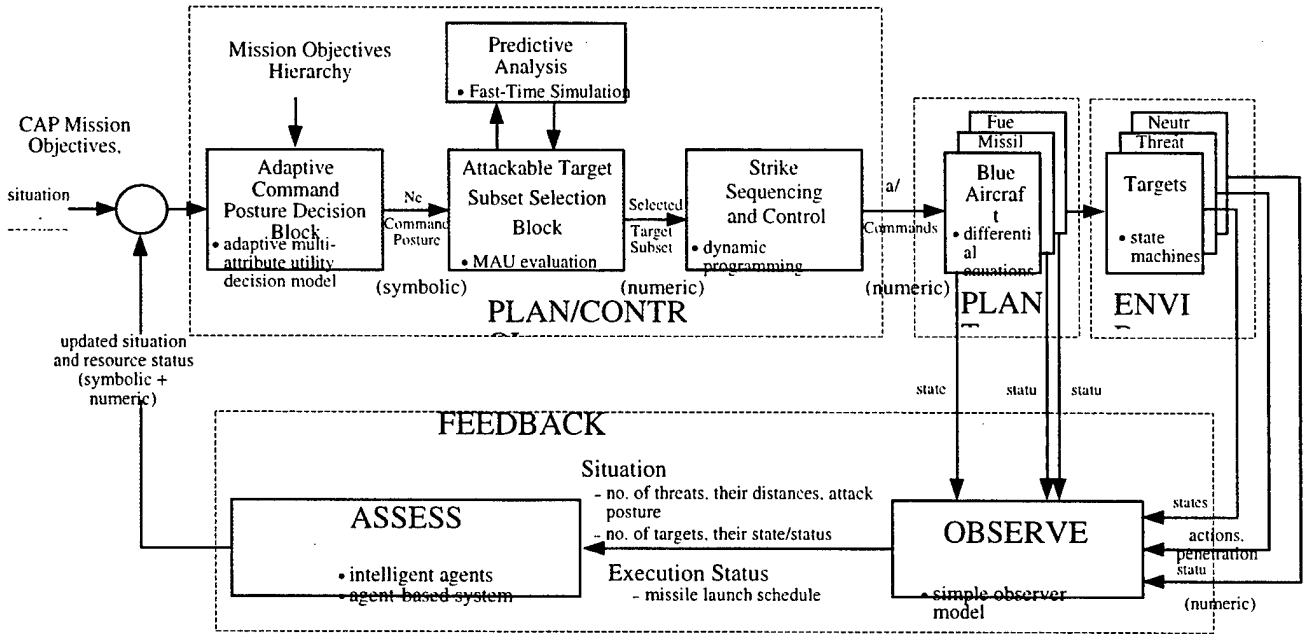


Figure 3. Symbolic Control Formulation of Fighter Aircraft Air Intercept Operations

The relative emphasis on the three key objectives shifts continuously depending on what the enemy is doing, what capabilities the Blue aircraft has, and the mission phase underway. Each objective can be further decomposed into explicit sub-objectives that are either directly measurable, or have measurable attributes associated with them. Each attribute provides a scale for measuring the degree of attainment of the associated sub-objective. The weighted combination of these attribute levels provides an indication of the attainment of each parent key objective. The weighted combination of the level of attainment of each key objective provides a measure of the overall attainment of the mission success objective (Figure 4). To model such an objectives hierarchy in which the objectives act in tradeoff fashion, use of hierarchical adaptive multiattribute utility (MAU) models [11] can be used. In this formulation, each key objective that acts in tradeoff fashion with the others is successively decomposed into sub-objectives to the point where the lowest level (terminal) objectives in the hierarchy are operationally measurable. With proper weighting of objectives in the objective hierarchy it becomes possible to measure the attainment of each low level objective and compute a weighted aggregate which yields a measure of the attainment of the overall mission goals.

Mission Command Postures. The relative weighting of the various attributes in the mission success hierarchy varies according to the tactical situation. A total of six tactical command postures symbolically capture the key tactical situations. Each command posture has a distinct set of attribute weights. The six command postures are:

(1) Offensive – maximize number of enemy downed, with secondary goals of maximizing carrier safety and resource conservation. ($W_2 >> W_1, W_3$) where W_1, W_2, W_3 are shown in Figure 4.

(2) Defensive – maximize-carrier safety, with secondary goals of maximizing E_k and resource conservation. ($W_1 >> W_2, W_3$).

(3) Conservative/Offensive – maximize E_k and resource conservation. Virtually ignore carrier safety. ($W_2, W_3, >> W_1$).

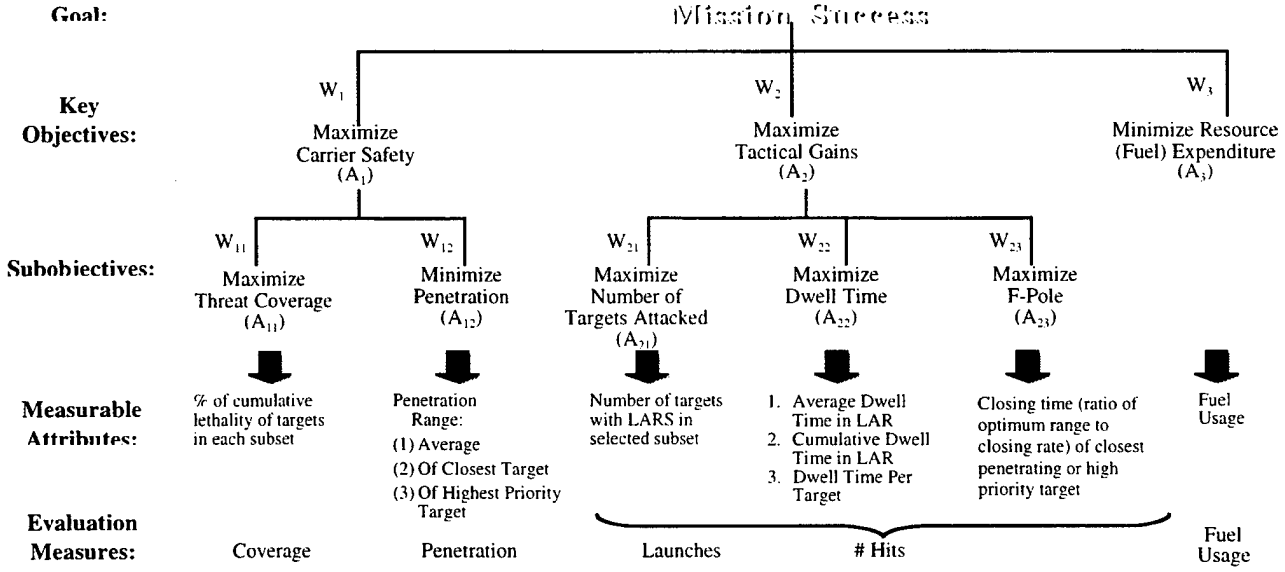
(4) Conservative/Defensive – maximize carrier safety and resource conservation. Virtually ignore E_k. ($W_1, W_3, >> W_2$).

(5) Carrier Safety – maximize carrier safety alone. ($W_1 = 1; W_2 = W_3 = 0$).

(6) Expected Kill (E_k) – maximize E_k alone. ($W_2 = 1; W_1 = W_3 = 0$).

The first four postures, Offensive (O), Defensive (D), Conservative/Offensive (C/O), and Conservative/ Defensive (C/D), are "tradeoff" strategies, i.e., they strike a tradeoff among the various objectives. Different combinations of attributes are emphasized in each. In the offensive posture, for instance, E_k is emphasized at the expense of carrier safety and resources. The final two postures, Carrier Safety and E_k are "pure" strategies. Carrier safety puts zero weighting on E_k and resources. E_k only weights the E_k attributes.

The initial conditions for this scenario are characterized by the receipt of mission guidance, initial command posture (e.g., conservative/offensive) current situation (i.e., target detection, number of targets, number of threats, their types, mutual support platform(s), etc.), available resources (i.e., full complement of long-range, medium range, and short-range missiles), and sufficient fuel to conduct air intercept operations and return to the CAP orbit.



Source: Madni and Freedy (1981)

Figure 4. Mission Success Hierarchy and Evaluation Metrics

Adaptive Command Posture Decision Block

The Adaptive Command Posture Decision Block dynamically updates the weights assigned to the mission objectives in light of the feedback from the Assess Block. The new mission command posture corresponds to the revised weights associated with the different objectives. The following paragraphs describe the transition logic and conditions associated with the tactical command postures.

The six command postures correspond to distinct tactical situations, characterized by the conditions or state of the following tactical variables:

(1) *Threat penetration range.* The threat distance from (a) the task force center or (b) the weapon release line around the carrier.

(2) *Fuel remaining.* The amount of fuel left to return to the carrier.

(3) *Numerical advantage.* The number of missiles compared to the number of targets.

(4) *Lethality.* Phoenix missile-equivalents of the threats.

An exhaustive set of relations between the postures (P_i) and the conditions (C_j) is given in Table 3. For example, defensive posture (P_2) is called for when high threat level exists, sufficient fuel remains, either a numerical advantage or disadvantage exists, and low threat lethality is low. Similar criteria for the choice of the other postures are presented Table 3.

Table 3. Command Posture Transition Logic

Command Posture		C_1	C_2	C_3	C_4	Transition Logic
P_1	Offensive	0	0	1	1	If $(P_1 \wedge (C_1 \wedge C_2) \wedge ((C_3 \wedge C_4) \vee C_3)))$ Then $\overline{P_1}$
		0	0	0	–	
P_2	Defensive	1	0	–	0	If $(P_2 \wedge (C_1 \wedge C_2 \wedge C_4))$ Then P_2
P_3	Conservative /Offensive	0	1	1	–	If $(\overline{P_3} \wedge (\overline{C_1} \wedge C_2 \wedge C_3))$ Then P_3
P_4	Conservative /Defensive	0	1	0	–	If $(\overline{P_4} \wedge C_2 \wedge ((\overline{C_1} \wedge \overline{C_3}) \vee C_1))$ Then P_4
		1	1	–	–	
P_5	Carrier Def. Only	1	0	–	1	If $(\overline{P_5} \wedge (C_1 \wedge \overline{C_2} \wedge C_4))$ Then P_5
P_6	Max E_K	0	0	1	0	If $(\overline{P_6} \wedge (\overline{C_1} \wedge \overline{C_2} \wedge C_3 \wedge \overline{C_4}))$ Then P_6

Conditions

C_1 = 1, enemy poses high threat (≥ target close to WRL)

$\overline{C_1}$ = 0, enemy poses low threat

C_2 = 1, fuel critical (inadequate fuel to complete engagement and return to carrier;

$\overline{C_2}$ = 0, fuel not critical

C_3 = 1, numerical advantage (onboard missile load ≥ no. of targets);

$\overline{C_3}$ = 0, numerical disadvantage

C_4 = 1, cumulative lethality (subset lethality $L \geq L_{min}$)

$\overline{C_4}$ = 0, low lethality

Source: Madni and Freedy (1981)

There are four tactical conditions which are monitored to determine posture transitions. These conditions are: C_1 – threat level, C_2 – fuel status, C_3 – numerical advantage status, and C_4 – cumulative lethality. The mapping of the tactical conditions to postures allows the automated transition from one posture to another as the sensed situation changes.

Attackable Target Subset Selection Block

An attackable subset is defined as that group of targets for which launch acquisition regions (LARS) exist for a specific location of the Blue fighter aircraft. Incremental changes in the aircraft location can result in a totally different attackable subset. An attackable subset can be no smaller than the onboard missile load if the number of targets is greater than the onboard missile load.

Evaluation of attackable subsets is accomplished by first forming all feasible combinations of targets with LARS. Feasibility demands that the number of targets in the subset is less than or equal to the number of missiles on board and that all geometric constraints are satisfied. Then each subset is assigned a vector of normalized attribute levels according to projected performance. The attributes are weighted by importance. The weighting is also normalized so that an overall zero implies zero on all attributes and an overall one implies a one on all attributes. The relative weighting differs according to command situation.

Strike Sequencing and Control Block

This block accepts the selected targets and creates a strike path through the target cluster. A major problem in conducting air intercept operations is optimizing the strike path and then executing the strike. This is an optimization problem that requires maximizing a performance measure related to the multi-target engagement in light of the raid structure and the operational capabilities and constraints of the aircraft. Tactical air problems are characterized by one key fact: control decisions taken at the present time affect the subsequent behavior of the system. Consequently, methods that provide a. solution in the form of a sequence of decisions over the entire duration of control, not just a decision at the present time are highly desirable. Dynamic programming offers a viable approach for solving optimization problems via a multi-state decisionmaking process. In essence, dynamic programming converts the simultaneous determination of the entire optimal control sequence into a more manageable sequential solution of vastly simpler intermediate optimization subproblems. The resulting solution is identical to that obtained by exhaustively searching all possible control combinations without performing the computationally prohibitive search.

Plant Models Block

The plant in this example is the Blue fighter aircraft with its onboard missile load and fuel supply. The degree of model fidelity is a function of the intent of the model. The Blue model should: (a) have the right outputs to support assessment and computation of measures of performance (MOPs); and (b) be able to "fly a route" that achieves the defined measures of effectiveness (MOEs). The latter implies being able to fit a "controllability window" or footprint defined by the onboard configuration and the aircraft "trajectory."

Environment Block

The Environment consists of the Opposing Force elements, and neutral elements. The Opposing Force consists of threats and targets. The key issue here is to identify those states that are needed for assessment and be able to output those observables. The level of fidelity needed in the Opposing Force model depends on whether the intent is to test the adequacy of the model for the assessment function or to improve the model. For the latter, it may be advantageous to output more observables than used in the assessment function to support "exploration." In our example, our interest is in observing the state of the targets (i.e., no damage, partially damaged, total kill, etc.), and the state of the threats (i.e., penetration range distance from weapon release line, etc.). Consequently, for assessment purposes, simpler models of the targets and threats can be used. Neutral elements can be ignored unless they become participants (i.e., a supporter or an agitator). If they do, they can be modeled as a "disturbance" that either help or hinder the efforts of the Blue Force (e.g., aircraft).

Observe Block

The observe block collects the situational information from the plant(s) and external environment as well as execution status of the plant. In our example, situational information include Red events (e.g., Red aircraft approaching weapon release line, new high priority threat identified), and Red actions (e.g., Red launch). Similarly, execution status pertains to Blue aircraft, state vector, and status relative to the "controllability window," missiles fired, targets remaining, and fuel remaining.

Assess Block

The assess block is responsible for interpreting the observations in terms of: their implications on goals, identification of opportunities, conflicts and constraints, evaluation of the consequences of the previous commands, and evaluation of existing conditions (e.g., threat penetration range, fuel remaining, numerical advantage status, and lethality) is analyzed by the assess block. The results are mapped to symbolic feedback which is provided to the Dynamic Tactical Command Posture Decision Block.

8. Illustrative Example: Enforce Cease Fire

This next example is at a higher echelon in the C^2 enterprise hierarchy. The intent here is to show that key objectives and constraints represented as symbolic variables can be mapped to numeric variables at this level.

Scenario: There is civil unrest in a Baltic country similar in size and location to Kosovo. The opposing elements of the population are approximately equal in size and widely inter-mixed. The active armed elements of each side are small and have been engaging in what amounts to widespread gang warfare throughout the country. The sporadic street battles threaten to polarize the populace completely and the government, while not taking sides openly, cannot contain the growing violence. The government has requested UN forces to enforce a cease fire while an attempt is made to reach some political accommodation. The US has committed to support the UN with military forces and has asserted a leadership role in defining detailed courses of action. The following paragraphs present exemplar objectives, constraints, and examples of symbolic to numeric mappings. For the exemplar scenario, the goals are to:

(1) *Reduce the level of violence and prevent clashes by armed groups*. Level of Violence (a symbolic variable) can be mapped to numeric variables as follows. Level of Violence can be described by the number of reported incidents, casualties, size and number of weapons used, interval between incidents, and number of people involved.

(2) *Sustain the authority and control of the situation by the local government and their indigenous military and police forces*. The level of Authority/Control, a symbolic variable, can be described by a weighted combination of the size of military and police forces, level of training, response time, frequency of direct intervention required as opposed to presence to resolve situation, approval rating by population in polls, number of volunteer recruits, percentage of favorable press reports, number of attacks directly on military/police as opposed to factions, percent of territory under effective control, military/police personnel to population ratio required to stabilize an area.

(3) *Prevent influx of arms from other nations or groups*. Arms influx, a symbolic variable, can be mapped to numeric variables such as number of shipments, number of suppliers, size of shipments, size and lethality of weapons.

The constraints on achieving the aforementioned objectives are:

(1) US domestic support for the activity is limited – the greater the support, the more money, troops and time can be spent on the operation. Domestic support is a function of perceived importance to US, economic impact [effect on markets or trade], perceived likelihood of success, possibility of US casualties, support of allies, viability of the local government.

(2) UN Funding is limited. ($ amount).

(3) Conditions are expected to be beyond recovery in six weeks (Schedule driver). Prospect of success is a function of degree of control, amount of arms influx, level of violence, popular support, etc.

(4) International support is mixed. US involvement must not appear to favor either side. Perceived balance of US involvement is a function of divergence of number of interactions with each faction from the mean, opinion poll results, balance of complaints from supporters of each faction.

In the interest of preserve the interests of the Blue forces, these constraints can also be differentially weighted.

The metrics or Measures of Merit of the Overall outcome are: Number of armed clashes. Level of violence in clashes. Percent of popular support for militant factions. Amount of outside arms. Amount of outside political influence. Percent of popular support for local government.

9. Experimentation Hypotheses

Table 4 presents the experimentation hypotheses, the accompanying rationale, experimentation process, and implications for the design of the symbolic controller.

The experimentation hypotheses are concerned with making the definitions of properties such as controllability and stability more precise, defining threshold conditions on various variables to assure stability, and developing mechanisms/heuristics to deal with contingency situations such as loss of feedback, erroneous human inputs, delayed resource availability.

Sample metrics include: a) ability to adapt to the perturbation; b) correctness of the adaptation; c) timeliness of the adaptation, i.e., ability to complete the adaptation or response prior to the occurrence of the next key event; d) ability to maintain stable operations in the face of perturbations/ disturbances described above; and e) ability to verify reachability of the desired end state from the current state.

Table 4. Experimentation Hypotheses for Symbolic Control and their Implications

	Hypothesis	**Rationale**	**Experimentation Process**	**Implications for Symbolic Controller Design**
Transformations	Mapping symbolic to numeric representation, analyzing/ processing in numeric domain, and mapping back to symbolic representation is possible for C^2 enterprise control	Required for symbolic control	Sample scenarios can be used to validate this hypothesis	Meaningful, consistent, and accurate SNST transformations are key to achieving symbolic control
Controllability	Controllability concept can be made precise for assessment and control	Needed for mission critical applications	Test scenarios will be designed to explore controllability regimes	Ensure that system does not go beyond controllability regime
Stability	Stability can be defined as a set of acceptable states of the plant	Acceptable states implies predictability and boundedness	Test scenarios to make definition of acceptable states precise; perturbation analysis	Boundedness of response is key to stable control
Performance	Performance index specific to situation and desired goal can be defined	Need performance metrics to assess system performance under nominal/degraded conditions	Use SNST approach to create mapping between symbolic and numeric performance metrics	In-situ performance assessment is required for real-time adaptation/intelligent control
Robustness	The controller is capable of adapting to certain types of perturbations	Control system needs to be robust with respect to changes in plant parameters, inputs, external disturbances	Test cases designed to verify and assess robustness of the controller (robustness margins)	Robustness is key to stability and performance
Time Delays	Time delay bounds for stability and performance can be defined	Need to characterize time delay bounds for stability and acceptable performance	Test cases designed to assess the effects of time delay on stability and performance	Time delays have to be small relative to system dynamics for stability and performance

Human Intervention	Human intervention situations should be pre-defined to ensure stable operation	Interventions in ongoing processes at arbitrary points can induce oscillatory behavior	Test cases with pre-defined entry points	Need to insure stable operations

10. Conclusions

This paper has presented a symbolic control formulation for a C^2 enterprise that: (a) frames the C^2 enterprise control problem within an OODA loop-oriented control system construct; (b) extends control-theoretic ideas into the domain of symbolic control; and (c) leverages multiple techniques from different disciplines within the overarching control-theoretic framework. Using suitable examples, the paper shows that control-theoretic concepts can be extended into the symbolic control domain through a set of meaningful and consistent transformations that involve symbolic-to-numeric mapping, numeric computation, and numeric-to-symbolic mapping. The paper defines key properties such as stability and controllability for the symbolic control domain and suggests key hypotheses for experimental validation of the key concepts and refinement of these definitions.

The illustrative examples are meant to convey two key concepts. First, mapping symbolic variables to numeric variables is feasible at both lower levels and higher levels in the C^2 enterprise. Second, it is possible to harness and combine techniques from different disciplines to create a complete solution that cannot be tackled by any one discipline alone.

These examples are meant to convey key insights and provide a point of departure for experimenting with different combination of techniques directed to building scaleable, symbolic control systems for complex enterprise control.

Acknowledgement

The author would like to express his appreciation to Professor Petros Ioannou of the University of Southern California and Mr. Jeffrey Koehler of Northrop Grumman Corporation for their helpful comments during the writing of this paper.

REFERENCES

[1] Madni, A.M. "Intelligent Process Management for C⁴ISR Enterprises," Intelligent Systems Technology, Inc. Internal Report, ISTI-IR-7/98-1, July 30, 1998.

[2] Madni, A.M. and Gayton, J.P. "A Process-Centric Crisis Action Planning and Execution System." Proceedings of the Thirteenth International Conference on Systems Engineering, Las Vegas, Nevada, August 10-12, 1999, pp. SE205-210.

[3] Boyd, John R., "Essence of Winning and Losing," unpublished briefing, 1996.

[4] J. Raisch, Control of Continuous Plants by Symbolic Output Feedback, Hybrid Systems II, Eds. P. Antsaklis, W. Kohn, A. Nerode and S. Sastry, Springer-Verlag, 1995.

[5] M. Zhou, D. T. Wang and D. Y. Chao, "Design of Command and Control Systems using Petri nets and Object-oriented Technology", Proc. of IEEE Intl. Conf. On Sys., Man and Cyb., pp. 2538-2543, 1994.

[6] P. C-Y. Sheu, I. Mayk and D. R. Smith, "Object-Oriented C^2 Systems", Proc. of IEEE Intl. Conf. On Sys., Man and Cyb., pp. 749-754, 1991.

[7] Ioannou, P. and Sun, M. *Robust Adaptive Control*, Prentice Hall Pub., 1996.

[8] Ramadge, P.J.G. and Wonham, W.M., "The Control of Discrete Event Systems", Proceedings of the IEEE, vol. 77, no. 1, pp. 81-98, January 1989.

[9] Zveren, C.M., Willsky, A.S. and Antsaklis, P.J., "Stability and Stabilizability of Discrete Event Dynamic Systems", Journal of the Association for Computing Machinery, vol. 38, no. 3, pp. 730-752, July 1991.

[10] Madni, A.M., and Freedy, A. Decision Aids for Airborne Intercept Operations in Advanced Aircrafts, Proceedings of the International Conference on Cybernetics and Society, Sponsored by IEEE Systems, Man, and Cybernetics Society, October 26-28, 1981, pp. 224-234.

[11] Madni, A.M. and Freedy, A. Decision Aids for Airborne Intercept Operations in Advanced Aircrafts, Proceedings of the Seventh Annual Conference on Manual Control, JPL Publication 81-95, October 15, 1981, pp. 187-210.

[12] Madni, A.M. "Integrated Modeling Approaches in Advanced Cockpit Automation," Proceedings of the 1983 SAE Aerospace Congress & Exposition, Long Beach Convention Center, Long Beach, CA, October 1983, pp 831543.

Closed-loop, Hierarchical Control of Military Air Operations

William D. Hall
C.S Draper Laboratory, MS # 3F
555 Technology Square
Cambridge, MA 02139
Ph: 617-258-2416
Fax: 617-258-1799
whall@draper.com

Milton B. Adams
C.S Draper Laboratory, MS # 3F
555 Technology Square
Cambridge, MA 02139
Ph: 617-258-3185
Fax: 617-258-1799
adamsm@draper.com

Abstract

Our approach to closed-loop, hierarchical control of military air operations employs a distributed control architecture that addresses disturbances at multiple levels of a hierarchically decomposed planning and execution system to accommodate the near and far term impacts of those disturbances. This paper describes the method we use to decompose the enterprise-wide optimization problem employing the theory of large-scale decomposition in a way that addresses shared objectives and highly constrained resources. We illustrate the technique on a simplified strike planning problem.

This approach provides a mechanism to assign local autonomy for distributed command and control elements that is consistent with the objectives and constraints established by superior elements, thereby allowing the preservation of traditional command and control organizational structures where desired.

1 Overview

Military air operations require command and control of diverse forces distributed over large geographic areas. The geographic distribution coupled with the need for short decision cycle times requires an agile, distributed and collaborative command and control capability for effective dynamic tasking of strike packages, supporting logistics, and sensing and electronic warfare assets.

This paper describes a **rigorous** approach to decomposing and executing large-scale decision-making problems for dynamic environments that combines the theories of decomposition of large-scale optimization problems and distributed control. This enables the replacement of heretofore ad hoc approaches to decomposing this class of large-scale operational problems, resulting in a distributed system for which the problem-solving and decision-making within each distributed C^2element addresses enterprise-wide objectives. Employing this approach to decomposition both provides significant insight into the nature of the feedback required to close the loop around each of the control elements within the decomposed problem, and defines the dynamics of the interactions among the control elements in solving the enterprise-wide problem, including the objectives passed from superior elements to subordinate elements and the feedback/status passed from subordinates to superiors.

2 Technical Approach

The closed-loop ***optimal control*** for each of the distributed elements is formulated as a receding horizon optimal control problem with the following attributes

Optimization – a time varying "set point" control and associated time history of system state values are determined over a finite time horizon to optimize an objective function representing the desired system performance.

Control – a high rate "perturbation controller" will augment the set point commands to stabilize the operation of the system and to ensure that the state of the system tracks the trajectory associated with the set point in the presence of disturbances.

We hope to achieve two objectives:

(1) Provide a structure/framework for solving the air operations problem that resolves both resource conflicts across lower levels as well as allocates objectives to the lower levels (a "mixed" decomposition).

This should all be done in a way that results in autonomy on the part of the (succeeding) lower levels in solving their decoupled problems. There are human-in-the-loop considerations on how one maps the decomposed problem onto the human organizational elements that ultimately must be responsible for planning and execution. Our approach provides an indication of the type of negotiation (e.g., iteration) and associated information exchanges among the levels required in arriving at a good, overall solution.

(2) Use the data/information exchanges among problem solving elements prescribed by the decomposition to form the basis of the real-time

feedback when we "close-the-loop" on the decision making.

That is, in accommodating our inability to exactly model/formulate the problem due to the myriad of uncertainties and unknowns that prevail in a real warfighting situation, the sensitivity to those uncertainties will be reduced using feedback (the purpose of closing the loop in even the simplest of control systems). Thus, the solution will evolve over time in response to the sensed state as well as new objectives provided by command levels.

Thus we will base the system architecture and information feedback for a large-scale, closed-loop control/decision-making system on a multi-level decomposition of the air operations optimization problem.

Benefits: The decomposition via the theory of large-scale optimization ensures that enterprise-wide objectives are pursued, and enterprise-wide constraints are honored by every element of the hierarchical and distributed system. The control architecture ensures that feedback is employed to reduce the sensitivity of the system to disturbances, time delays and model uncertainties.

2.1 Introduction

The theory of large-scale optimization has principally addressed either static problems or open-loop solutions to dynamic problems that have been discretized in time to allow their formulation as essentially static problems. In contrast to ad hoc approaches to decomposition that are too often employed in problem-solving, this theory has led to proper decompositions of very large scale problems into components or subproblems that are computationally tractable as well as comprehensible by human decision-makers. The latter is important because the humans who are ultimately responsible for carrying out the solutions are more comfortable in that role if they are provided an intuitive grasp of the nature of the solution.

2.2 Development of the Decomposition

The plan generation process must be coordinated across adjacent levels of the command and control hierarchy in order to meet enterprise-wide objectives, and there must be an internal control of the interactions among levels in order to insure the stability of the overall plan generation process.

2.2.1 Technical Challenges: Decomposition

The following basic questions must be answered in developing decompositions for large-scale enterprise operations:

- How many levels are required?
- How should problem solving be partitioned across levels?
- What constraints and objectives should be passed from level to level?
- What is the nature of the status that is passed from subordinate to superior levels?
- How is problem-solving best accomplished across levels?
- What happens when a level cannot meet its objectives and/or honor its constraints?
- To what extent should the decomposition reflect human-system interaction concerns?
- How might one develop a system wherein levels are established dynamically?

Analytical approaches to decomposing *Large-scale Optimization Problems,* and the essential role played by subproblem coordination (which is itself formulated as an optimization problem) have been developed over the last three decades [1,2,3]. The formal analytical developments help to establish methodologies for achieving decompositions wherein the subproblems are properly coordinated via a higher *Master* level. These approaches have been extended in the development of methodologies for the decomposition of *Large-scale Control Problems* for linear dynamic systems and quadratic cost functions [4, 5]. The central topic of our effort is the extension of this methodology to closed-loop control for large-scale enterprise problems with more complex objective functions and constraints.

2.2.2 General Approach To Multi-Level Optimization

The objective of multi-level optimization is to decompose a complex optimization problem into a hierarchy of simpler problems. The simpler optimization problems are solved independently at each level of the hierarchy, with the superior or master levels coordinating the solutions of the decoupled subordinate level problems. The discussion below of multi-level decompositions is intended to be qualitative. Technical details and conditions can be found in the references cited above.

Consider the typical problem statement:

$$\min_{\underline{x},\underline{y}} f(\underline{x},\underline{y})$$

subject to $\quad \underline{g}(\underline{x},\underline{y}) \le \underline{0}$

where $\quad \underline{x} = \begin{bmatrix} \underline{x}_1 \\ \underline{x}_2 \\ \cdot \\ \cdot \\ \underline{x}_N \end{bmatrix}$

The vector $\underline{x}$ is composed of subvectors $\underline{x}_i$ which will later be associated with the N subproblems at a lower, subordinate level of the decomposition. The vector $\underline{y}$ corresponds to the variables that couple the subproblems in either or both the objective function f or the constraint vector g. The problem can be rewritten in terms of a Lagrangian L with Kuhn-Tucker multiplier vector $\underline{\gamma}$:

$$L(\underline{x},\underline{y}) = f(\underline{x},\underline{y}) + \underline{\gamma}^T \underline{g}(\underline{x},\underline{y})$$

The decomposition of the optimization problem is achieved by creating a superior or master level problem whose solution produces the value of the coupling vector $\underline{y}$ or the multiplier $\underline{\gamma}$. The former is referred to as "interaction prediction" and the latter as "goal coordination" or "price coordination." To establish the decomposition and define the superior and subordinate problems, the Lagrangian is rewritten as a sum of decoupled Lagrangians (i.e., setting a value for $\underline{y}$ leads to separability of both the objective function f and the constraint vector g):

$$L(\underline{x};\underline{y}) = \sum_{i=1}^{N} [f_i(\underline{x}_i;\underline{y}) + \underline{\gamma}_i^T \underline{g}_i(\underline{x}_i;\underline{y})]$$

$$= \sum_{i=1}^{N} L_i(\underline{x}_i;\underline{y})$$

In this case, given values for $\underline{y}$ each of the N subordinate levels is responsible for solving a decoupled optimization problem associated with one of the decoupled Lagrangians, L_i:

$$\min_{\underline{x}_i} f_i(\underline{x}_i;\underline{y})$$

$$\underline{g}_i(\underline{x}_i;\underline{y}) \le \underline{0}$$

Iterations between the upper and subordinate levels ***are required to achieve an optimal solution***. The nature of the iterations is a direct by-product of the decomposition.

The variables $\underline{x}_i$ might represent coordination points or times that couple adjacent mission phases, where the i_{th} subproblem would be the planning problem for the i_{th} mission phase. For a mission involving multiple strike packages these variables might be parameters that enforce coordination across those missions. In that case, the i_{th} problem would be the mission for the i_{th} strike package. Of course those missions would be further decomposed for the individual aircraft comprising the strike package.

The formal details of the decomposition for the price coordination method depend on the nature of both f and g. For some problems, price coordination alone may be insufficient to completely decouple the problem. In general, the objective is to achieve unconstrained subordinate level problems of the form:

$$\min_{\underline{x}_i} f_i(\underline{x}_i) + \underline{\gamma}_i^T \underline{g}_i(\underline{x}_i)$$

where the nature of the problem has allowed us to avoid dependence on the coupling variables $\underline{y}$. The spirit of price coordination is that the superior or master level does not explicitly set values for the coupling variables but sets "prices" or penalties (i.e., values for the multipliers) for violating the constraints. As with the interaction prediction approach, iterations between the upper and subordinate levels are required to achieve an optimal solution.

The price coordination and interaction prediction approaches can be combined ("mixed") wherein the superior or master level fixes values for any subset of the coupling variables that is sufficient to decouple the Lagrangians, and sets prices for constraint violations that are evaluated at the fixed values of the coupling variable $\underline{y}$.

2.3 Closing the Loop

The solutions to the subproblems at the lowest levels of the decomposition represent a plan of activities that are to be pursued by the enterprise's physical entities in prosecuting the business of the enterprise, e.g., missions for individual aircraft. At higher levels, the solutions produce objectives and constraints to be employed by successively lower levels, e.g., allocation of sets of targets to sets of strike packages. The environment (the plant) within which those activities are to be pursued is represented (modeled) in the formulation through a variety of constraints. Of course, the plant model will not completely and accurately predict the state of the plant at future times. One can view differences

between the plant model used to predict future states in the problem formulation and the evolution of the actual plant state as being attributed to disturbances. In order to reduce sensitivity to disturbances, we will employ feedback about the actual evolution of the state. Indeed, the earliest designers of control systems realized that open loop solutions would quickly diverge due to disturbances, and thus, they employed feedback. In the following we discuss the general architecture that we use in closing the loop around complex, distributed enterprise command and control systems.

2.3.1 General Closed-Loop System Architecture

As discussed earlier, solutions to complex problems are made tractable by decomposing them into simpler, decoupled subproblems that can be solved (nearly) independently. In order to reduce sensitivity to modeling errors and other types of disturbances, feedback is employed. Figure 1 is a general representation of one of the command and control elements within the decomposition. Feedback is provided by sensing the "system to be controlled." The "system" may be physical entities within the plant that are being controlled or it may represent an aggregation of lower level problem solving elements along with the entities they control. A closed-loop, hierarchical decomposition is a recursive implementation of the functional decomposition illustrated in Figure 1, where the "system-to-be-controlled" is one or more subordinate level processes that are "controlled" or coordinated by an upper Master level as shown in Figure 2.

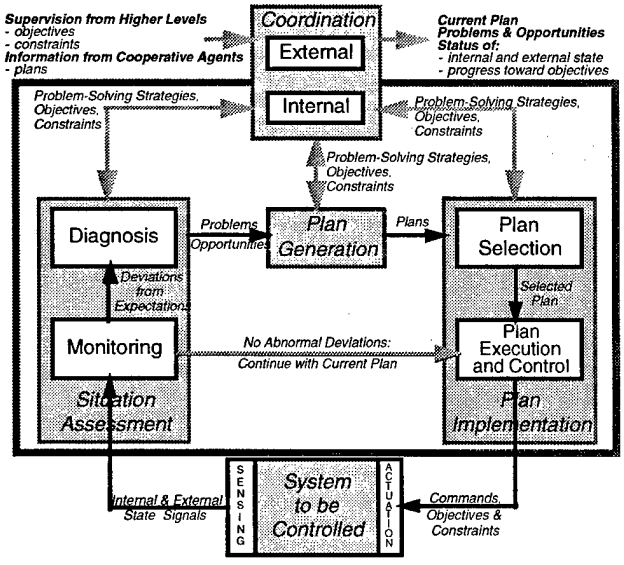


Figure 1: Functional Decomposition of a Command and Control Element

The nature of the feedback required is a function of the nature of the subproblems to be solved. The feedback should contain the information required to evaluate progress toward the solution to the subproblem being solved. Since the solutions generated will span a finite time horizon, models will be required to estimate/predict future states/status using current state information.

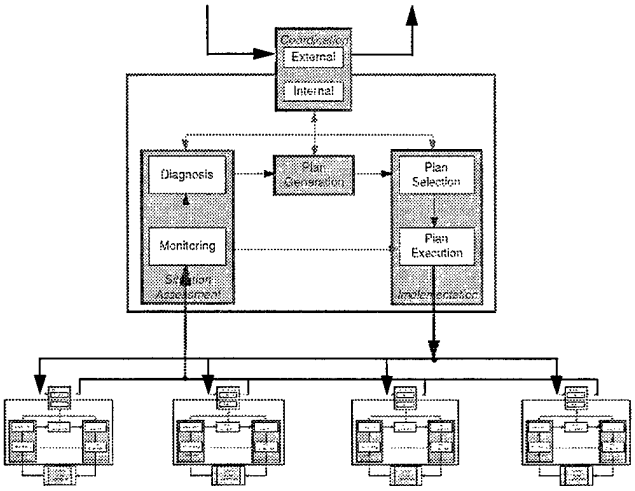


Figure 2: Hierarchical View: Aggregated Plant for Master Level

In Figure 2, the plant for the Master level is an aggregation of the lower level elements and their plants.

3 Problem Formulation & Decomposition

3.1 Strike Planning Problem

In this section, we illustrate the decomposition techniques outlined above by applying them to a simplified strike planning problem that is based on the work in the theses [6, 7]. In this strike planning problem, aircraft and weapons are tasked to strike targets of interest. Each aircraft is based at one of a number of bases within range of the targets.

Aircraft are tasked together in groups called strike packages. The aircraft in a strike package work together to accomplish a mission objective. Different aircraft in each strike package typically perform specialized functions that contribute to the overall effectiveness of the package. For instance, a strike package may consist of bombers escorted by fighters. The fighters in the strike package engage enemy air patrols that could hamper the bombers' ability to reach their targets. It is assumed in this example that the strike package assigned to a target consists of aircraft and crew based together at the same location, allowing a common pre-mission briefing.

package. The upper level coordination addresses these issues, ensuring that important targets will not be ignored due to resources being wasted on the multiply-hit target..

We decompose the problem according to two of the approaches described earlier. The first decomposition is of the interaction prediction, or feasible type. The second decomposition is of the price-coordinated, or infeasible type.

3.3.1 Interaction Prediction Decomposition

To decompose the problem via interaction prediction, we introduce dummy variables σ_{ij} which are constrained to be equal to the y_{ij}. The master problem will set the σ_{ij}, which the subproblems will use to coordinate their solutions. The formulation is presented below, augmented with the σ_{ij} in equation 10.

$$\text{maximize}_x \quad \sum_i \sum_j \sum_n PREF_{jn} x_{ijn} \tag{6}$$

$$\text{subject to} \quad \sum_i \sum_n x_{ijn} \le 1, \ \forall j \tag{7}$$

$$\sum_n \sum_j x_{ijn} QTY_{ajn} \le AVAIL_{ai}, \ \forall a, i \tag{8}$$

$$y_{ij} = \sum_n x_{ijn} \tag{9}$$

$$y_{ij} = \sigma_{ij}, \ \forall i, j \tag{10}$$

$$x_{ijn} \in \{0.1\}, \ \forall i, j, n \tag{11}$$

We can dualize the constraint on the σ_{ij} (equation 10) to produce the following equivalent problem.

$$\text{minimize}_\lambda \ \text{maximize}_x \quad \sum_i \sum_j \sum_n PREF_{jn} x_{ijn} + \lambda_{ij}(\sigma_{ij} - y_{ij}) \tag{12}$$

$$\text{subject to} \quad \sum_i \sum_n x_{ijn} \le 1, \ \forall j \tag{13}$$

$$\sum_n \sum_j x_{ijn} QTY_{ajn} \le AVAIL_{ai}, \ \forall a, i \tag{14}$$

$$y_{ij} = \sum_n x_{ijn} \tag{15}$$

$$x_{ijn} \in \{0, 1\}, \ \forall j, n \tag{16}$$

This problem can now be decomposed by interaction prediction. If we take y, x, and λ to be given since they are solved by the subproblem, the above program simplifies to:

$$\text{maximize}_\sigma \quad \sum_i \sum_j \sum_n PREF_{jn} x_{ijn} + \lambda_{ij}(\sigma_{ij} - y_{ij}) \tag{17}$$

$$\text{subject to} \quad \sum_i \sigma_{ij} \le 1, \ \forall j \tag{18}$$

$$\sigma_{ij} \in \{0, 1\} \tag{19}$$

The solution technique employed to solve this master level problem must recognize the fact that y and λ are functions of σ. That is, σ should be updated so that the solution converges; if the master level problem is solved treating y and λ as fixed constants at each iteration, the procedure is likely to cycle.

Given a value for σ, the subproblems solve for y, x, and λ. The resulting program for base subproblem i is:

$$\text{minimize}_\lambda \ \text{maximize}_{x,y} \quad \sum_j \sum_n PREF_{jn} x_{ijn} + \lambda_{ij}(\sigma_{ij} - y_{ij}) \tag{20}$$

$$\text{subject to} \quad \sum_n \sum_j x_{ijn} QTY_{ajn} \le AVAIL_{ai}, \ \forall a \tag{21}$$

$$y_{ij} = \sum_n x_{ijn} \tag{22}$$

$$x_{ijn} \in \{0, 1\}, \ \forall j, n \tag{23}$$

The subproblems thus determine not only how to strike their allocated targets, but through λ communicate the sensitivity of their solutions to assignment of additional targets or removal of existing target assignments. This information is used in the following iteration at the master level to derive the next σ.

3.3.2 Price Coordinated Decomposition

For price-coordinated decomposition, we do not need the σ dummy variables. Instead, we start with the original formulation and dualize the coordinating constraint (equation 2). The result is:

$$\min_\mu \max_x \quad \sum_i \sum_j \sum_n PREF_{jn} x_{ijn} + \sum_j \mu_j \left(1 - \left(\sum_i \sum_n x_{ijn}\right)\right) \tag{24}$$

$$\text{subject to} \quad \sum_n \sum_j x_{ijn} QTY_{ajn} \le AVAIL_{ai}, \ \forall a, i \tag{25}$$

$$x_{ijn} \in \{0, 1\}, \ \forall i, j, n \tag{26}$$

$$\mu_j \ge 0, \ \forall j \tag{27}$$

In the price-coordinated decomposition, the μ_j represent the value that some subproblem can achieve by hitting target j. These prices are set by the master level according to the minimization:

$$\text{minimize}_\mu \sum_j \mu_j \left(1 - \left(\sum_i \sum_n x_{ijn}\right)\right) \tag{28}$$

$$\text{subject to} \quad \mu_j \ge 0, \ \forall j \tag{29}$$

The solution method employed at the master level must recognize that x is a function of μ. The master level problem is updated at each iteration using the results of the subproblems in the previous iterations.

Given the μ, each base subproblem i solves the following maximization problem:

$$\text{maximize}_{x_i} \quad \sum_j \sum_n PREF_{jn} x_{ijn} + \mu_j \left(1 - \left(\sum_n x_{ijn}\right)\right) \tag{30}$$

$$\text{subject to} \quad \sum_n \sum_j x_{ijn} QTY_{ajn} \le AVAIL_{ai}, \ \forall a \tag{31}$$

$$x_{ijn} \in \{0, 1\}, \ \forall j, n \tag{32}$$

Thus, subproblem i must effectively justify its choice to hit target j by paying (through the penalty term in the objective function) the dual cost . If the dual cost is too

great, that signifies that another base is better able to hit the target, so base *i* does not include it in its solution. If, on the other hand, base *i* is able to hit the target better than other bases, it will at the optimal solution include target j and generate more value in so doing than it must pay through the dual cost.

Expected Results

Once our formulation has been completed, implemented and embedded in a closed-loop system such as that depicted in Figures 1 and 2, we will investigate the following: (a) speed of convergence, (b) nature of the information exchanged between superior and subordinates, and (c) human considerations. The speed of convergence is important because the solution process is embedded within a closed-loop system, and any reduction in the transport lag attributable to control law computation will improve both robustness and stability of the closed-loop performance. The complexity of the information exchanged between superior and subordinate during the solution process can have a similar impact on transport lag. The human considerations relate to the humans' ability to gain insight into the solutions developed for the individual subproblems. In particular, if humans are not able to grasp intuitively the solutions to the subproblems, then the human oversight required in monitoring the execution of those solutions may be jeopardized.

References

1 Singh, M. H. and A. Titli, Systems: Decomposition, Optimization and Control, Pergamon Press, Oxford, 1978.

2 Tamura, H. and T. Yoshikawa, eds., *Large-Scale Systems Control and Decision-Making*, Marcel Dekker, New York, New York, 1990.

3 Lasdon, L. S., *Optimization Theory for Large Systems*, Macmillan, New York, New York, 1970.

4 Mahmoud, M.S, Hassan, M.F. and Darwish, M, G., *Large-Scale Control Systems: Theories and Techniques*, Marcel Dekker, New York, 1985.

5 Drouin, M., Abou-Kandil, H., and Mariton, M., *Control of Complex Systems*, Applied Information Technology Series, Plenum Press, NY, NY 1991.

6 Dolan, Matthew H., "Air Tasking Order (ATO) Optimization Model", MS Thesis, Department of Operations Research Naval Postgraduate School, September 1993.

7 Crawford, Kevin R., "Enhanced Air Tasking Order Optimization Model", MS Thesis, Department of Operations Research Naval Postgraduate School, September 1994.

The Situation Assessment Problem: Toward a Research Agenda

Alexander Kott, PhD,
Logica Carnegie Group
kotta@logica.com

Martha Pollack, PhD,
University of Pittsburgh
pollack@cs.pitt.edu

Bruce Krogh, PhD,
Carnegie Mellon University
krogh@ece.cmu.edu

1. Introduction

This paper presents the results of a short study on the nature and challenges of the "Assessment Problem," and the approaches applicable to the solution of the problem. This study was performed as a task within the DARPA's JFACC project, in February-March of 1999, primarily in several brainstorming sessions conducted by the authors.

2. Motivation

As used in this paper, the "Assessment" is a task within the military Command and Control process, in which the data obtained from the battlespace are transformed into information that describes how the state and events of the battlespace differ from the expectations (explicit or implicit) of the Command decision-makers. This definition will be elaborated and discussed shortly.

There are several motivations for studying this problem. Practical concerns and needs of military professionals are the most important ones. The need to assess the battlefield situation and to answer questions like "what's going on?" and "how are we doing?" has always been a key challenge of a commander's job. However, the Information Revolution has dramatically exacerbated this challenge. The amount of data potentially relevant and available today to a Commander of a major military operation vastly exceeds any human's ability to review, comprehend and understand these data. Military decision makers face an immediate need for assistance in the job of transforming enormous amounts of low-level data, incomplete, uncoordinated and uncertain, into a few aggregated, understandable and actionable elements of information. These products must be available in a manner that is timely and relevant to the given Commander's functions and current concerns.

Today's military operations suffer from a feedback mechanism that does not support reliable performance assessments, timely responses to information requests, consistent situational analysis, or relevant reporting factors. In other words, many decisions are made without the benefit of the disciplined incorporation of feedback on operations. Although a significant amount of feedback data, such as Battle Damage Assessment, number of sorties flown, number of targets serviced, etc., are available, the feedback does not support the commander and staff in a reliable, timely, consistent, or relevant fashion, nor at the level of abstraction and integration that allows decision makers to make effective decisions.

Although similar assessment problems exist in other fields of human endeavor, e.g., in management of commercial enterprises, the military C^2 domain adds a number of unique challenges. The sensor data are inherently uncertain and incomplete, being affected by the nature of the environment and also by the intentional actions of the adversary intended to deceive and confuse the friendly information collectors. The volume of observed data and the rate of change are extremely high, however only a fraction of the available data are actually relevant at any given time. Therefore, the placement of sensors, and the importance of the available data must be dynamically determined and frequently changed depending on the situation. Decisions often depend on a pattern of observed data that may emerge anywhere and anytime and may not be predicted a priori. Assessing the situation includes determination of friendly characteristics as well as those of the adversary.

Motivations and many aspects of the Assessment Problem will look at least partially familiar to many researchers and technologists. From the perspective of the researchers in the field of Planning, the Assessment Problem shares much with the Execution Monitoring problem and the recent exploration of the sentinel concept. From the perspective of control theory researchers, the military Command and Control can be viewed as a control system in which the feedback loop involves sensors/observers, interpretation and computation of differences between the reference vector and the feedback.

3. The Method and the Scope of This Study

We performed this very short, limited and focused study in the following manner. The small team (the authors of the paper) was selected to combine the expertise in areas that were thought most pertinent: development of tools for decision-making in C2 environments; enterprise control; dynamic planning. The team iterated through the following steps: formulation of the subproblems and their challenging characteristics; identification of potentially applicable research fields; potential contributions of each candidate field; limitations of each candidate research field from the perspective of the Assessment Problem and its challenges.

To limit the scope of this study, we had to exclude a large number of topics that one could justifiably consider highly relevant to the Assessment Problem. For example, we excluded the issues and research related to raw sensor interpretation, sensor fusion and natural text (e.g., natural text interpretation). We assumed that all Battlespace data are made available to the Assessment System in a structured form. We did not consider the deliberate assessment (forward evaluation) of plans of actions, COAs, etc.; instead we focused exclusively on real-time assessment that occurs while actions are executed.

4. Definition of the Assessment Problem

One way to define the scope of a problem is to postulate a hypothetical mechanism for solving the problem. Figure 1 depicts a hypothetical Assessment System as we defined it for the purposes of this study. The Assessment System receives a stream of Battlespace Data from the Collection System that in turn obtains these data from the Battlespace. The Assessment System also receives the continuously updated Order from the Command System. The Order describes the actions that the Command System wants to be accomplished in the Battlespace, along with their intent and expected effects. This Order does not prescribe the operations of the Assessment System itself, but it is an important reference information that enables the Assessment System to have the visibility into the intents and expectations of the Command System. Internally, the Assessment System continuously updates its set of Current Beliefs about the Battlespace and about the Collection and Command systems. It also continuously updates its Utility Function that defines the value of the Assessment System's output to the Command System. Another utility function – the External Utility Function is received from external source and serves as a fundamental value system or a modifier to the internally computed Utility Function. The Assessment System outputs the Collection Guidance – the request to collect certain information – to the Collection System. Finally, the most important output of the Assessment System is the Assessments - the deviation between the expected state of the world (where state is not necessarily the current state, but a sequence of states extending into the future) and the state estimated from the available information.

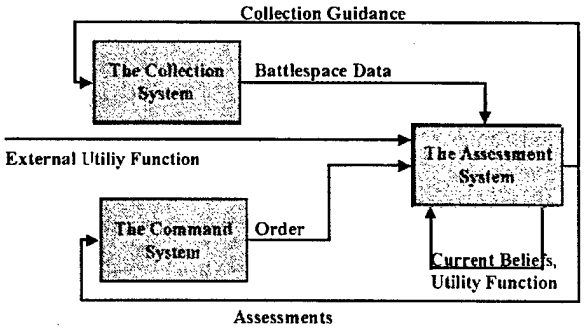


Figure 1. Information flows of the assessment system

Using Figure 1 as a guide, we suggest that the Assessment Problem can be decomposed into the following five sub-problems. The first of the five sub-problems is the primary problem – its solution is the main output of the Assessment System. The other four problems are auxiliary to the first one.

Problem 1. Compute the Assessments – the set of differences between the expected and the estimated actual state of the Battlespace, such that utility of the Assessments are maximized over a sequence of computations and the costs are minimized.

Discussion: In general, a state of the Battlespace is defined over a time interval of the interest to the Command system, i.e., a state is not a snapshot at a single moment in time, but a dynamic development of the Battlespace over time within the interval of interest. Expectations can be explicit or implicit, e.g., an appearance of a new threat within the Battlespace is a deviation from an implicit assumption about the known threats. The utility would typically include relevance to the Command System's current interests, timeliness of the assessments, and level of confidence in the assessments. The costs may include consumption of computational resources, and possibly the size of the Assessments set – greater size implies greater processing effort on the part of the Command System. In general, the Assessments represent only a small fraction of all differences that exist between the expected and actual state. It is that fraction that is most relevant to the interests of the Command System, most reliable, etc. All differences cannot be computed because of the limitations of both the available information and the computational resources. Furthermore, all differences should not be computed and

sent to the Command System because most of them are insignificant. The phrase "over a sequence of computations" stresses that utility of the Assessments should be consistent over time, e.g., it is not generally desirable to produce excellent assessments at moment t+1 at the expense of sacrificing the assessments at time t. The challenge is to continuously produce a right-sized set of highly relevant, reliable, timely assessments.

Problem 2. Compute the set of Focal Data – a subset of Battlespace Data – such that the use of the Focal Data for computing Current Belief and Assessments maximizes the utility of Assessments subject to constraints on computational costs and time available.

Discussion: In general, the entire available Battlespace Data will be too large for computing the Assessments within the cost and time constraints. The challenge is to focus on a smaller subset of data that would result in best assessments possible within the cost and time constraints. The key idea is that a system with bounded resources must be able to decide which data to consider and how to prioritize the system's attention when much is happening. It may also be useful to distinguish between two points at which focusing of attention is required. First, because it's infeasible to consider all available data, a decision must be made about what data to take into account. Second, when things start happening and data change rapidly, it may be impossible to attend to *all* that information and still produce timely assessment report. Thus, a second set of decisions must be made, about what information to focus on (first). In both cases, the decisions may be linked and dynamic, e.g., if you see X, start monitoring Y; if you see X, next focus on the interpretation of Z.

Problem 3. Compute the Collection Guidance that maximizes the utility of the resulting data for the future computations of Assessments, subject to cost constraints.

Problem 4. Compute Utility Function such that maximizing the Utility Function implies maximizing the expected fulfillment of the Order.

Problem 5. Compute the Current Beliefs about the state of Battlespace such that the use of the Current Beliefs for computing Assessments maximizes the utility of Assessments subject to constraints on computational costs and time available.

5. General Characteristics and Challenges of the Problem

In searching for approaches to solving the Assessment Problem, one should consider some of its key characteristics:

- The Battlespace data are very large, heterogeneous, dynamically changing datasets; therefore any approach or technique must be critically analyzed with a view toward its potential ability to scale up, to operate in near real time and to handle data dynamics.
- The Battlespace data will always be substantially incomplete and present only a part of the picture; potentially observable items far exceed any realistic capabilities and number of sensors.
- The hostile environment, in which the Battlespace data are collected assures that data available have a substantial inherent uncertainty, varied degree of latency, contradictions, and noise. These data are impacted by actively counteracting the enemy, by deception, and by intentionally created confusion.
- The Assessment System must account for the fact that the Battlespace itself is dynamically changing system where changes are not merely parametric but structural. It includes friendly and enemy forces that rapidly and continuously change their composition and relations to each other. Thus, for example, a model of the Battlespace cannot be a static one.
- The Orders of the Command System, including its goals and intents, also change dynamically and widely. This engenders yet another need for dynamic adaptation within the Assessment System.
- There are strong real-time requirements. Time-value of the assessments is a critical consideration: lower quality information now can be much more useful than higher-quality information a few hours or even minutes later.
- In Command Systems found in military domains, decision-makers are found at different echelons and different levels of authority. There is a broad range of abstractions at which potentially useful assessments can be made, depending on the role and the level of authority and responsibility of the decision-makers.
- In addition to multiple functions and echelons, individual tastes, preferences and command styles of the human decision-makers produce a strong impact on how an assessment should be performed. An Assessment System requires models of individual decision-makers, both automated and human.
- Most of today's information technologies are strongly dependent on models of the phenomena or systems that they serve or analyze. However, one must question the availability and applicability of models in the military domain, such as models of Battlespace and Command. Although certain aspects of weapons and forces have been extensively modeled, comprehensive models of Battlespace and Command do not exist. Perhaps even more importantly, one may question if any model of military matters, which by necessity are biased toward "the last war," will ever be applicable to "the next war." Perhaps we must look for model-independent approaches, or for means to

construct models automatically and dynamically. We will return to this point later.

6. Current Research Directions: Applicability and Gaps to Overcome

In this section we discuss a number of research fields from the perspective of the Assessment Problem. For each field we discuss (a) the potential contributions of this research into the Assessment Problem and (b) the limitations of the research that must be overcome or mitigated in order to apply it to the Assessment Problem.

Research field: Model-Based Diagnosis, e.g., [1], [2].

Potential contributions of this research to the Assessment Problem: approaches to inferencing the requirements for specific information from the decision-making model or from the system model

Current limitations of this research:

1. MBD research has not addressed the need for focusing on a specific most valuable subset of the available information.
2. The most significant similar current use of MBD is in Robotics and most work tends to be of ad hoc nature.
3. There is limited understanding of how MBD can be applied in real-time, resource-bound situations. This topic deserves a more detailed discussion. The work on Model-Based Diagnosis has made fairly significant progress in this respect: significant enough that it's being used in Deep Space One, which does involve a dynamically changing situation and real-time constraints [2]. However, getting MBD to work for DS1 involves an extensive amount of engineering; moreover the designers of the MBD component were also working closely with the systems engineers, so that they fully understood the performance of the system, and to some extent could control it. In a warfare situation, no side can have a complete understanding of the enemy behavior, and they certainly don't have much control over it in advance

Research field: Execution Monitoring, particularly the recent work on Rationale-Based Monitoring, e.g., [10]; [7], [9], [8].

Potential contributions of this research to the Assessment Problem: approaches to determining most valuable information to monitor

Current limitations of this research: Only very preliminary work has been done on this so far; further, it has so far only been tested as a technique for monitoring during the planning process, not for monitoring during execution. Although the latter is an intended extension of the idea, it hasn't been implemented and analyzed yet. Also, it hasn't addressed the question of links between different monitors (i.e., the "if you see X, then check Y" issue.)

Research field: Machine Learning and Data Mining, e.g., [21], [22], [23].

Potential contributions of this research to the Assessment Problem: Machine learning and data mining techniques provide a way to discover significant patterns in the massive amount of data that are being input to the assessment system. ML/DM techniques divide into three main classes: supervised, reinforcement, and unsupervised. In supervised learning, the system is given immediate feedback about what the "correct" answer was. In reinforcement learning, the system occasionally receives rewards or penalties, but not direct feedback about the correct answer. In unsupervised learning, the system must identify clusters of data that are similar in some way, without external input about the quality of its results. "Data mining" is a term used to describe the application of machine learning techniques to very large, typically distributed, databases, which may contain rich implicit regularities. One form of data mining that may be particularly relevant to assessment systems involves learning the structure of Bayesian networks. If Bayesian networks could be inferred from the incoming data, they could provide a significant amount of assessment information. ML/DM approaches are particularly attractive to the Assessment Problem because they are at least partially model-independent.

Current limitations of this research: Supervised learning may be problematic for the assessment problem, because it requires human input to provide correct results during training, but often the human will not understand enough about the situation to do so. (If we had this knowledge, we wouldn't need an assessment system at all: we could just have a human assistant perform the assessment.) Reinforcement learning could in principle address this problem, but reinforcement learning is computationally complex, and often doesn't converge until after a large amount of experience has been gained. In a highly dynamic environment, a reinforcement learning algorithm may not have time to learn a model before it has changed. The highly dynamic nature of the environment will also pose a challenge for unsupervised learning techniques; in addition, they do not address the real-time issues (i.e., they do not include explicit mechanisms for trading solution quality against computation time). Although in principle data-mining approaches have an important advantage of being model-independent, unless the system has some idea of what it is looking for, there are likely to be vastly too many patterns found in the data. Thus, some form of model is needed to focus on what's interesting, and identify "useful" information.

Research field: Aggregation of Dynamic Modes [24], [25]; [26]; [27].

Potential contributions of this research to the Assessment Problem: Techniques for creating reduced-order models reflecting the significant dynamics for assessment
Current limitations of this research: Principally based on off-line analysis of detailed analytical models.

Research field: Recursive State Estimation (Kalman Filters)
Potential contributions of this research to the Assessment Problem: On-line estimation of the quantified elements of the current state of the world
Current limitations of this research: strictly numerical, limited robustness results (robustness with respect to model inaccuracies; confidence measures depend on priors); highly model-dependent; issue of selective relevancy is not addressed.

Research field: Model Validation, e.g., [28].
Potential contributions of this research to the Assessment Problem: determining quality of models from on-line data and selecting which of the possible models of the system is applicable given the observations
Current limitations of this research: principally based on linear dynamic models (in the dynamic systems literature); alternative literature exists for discrete event simulation and training (training set / validation set concepts)

Research field: System Identification. There is a very rich literature in this field, and many successful applications.
Potential contributions of this research to the Assessment Problem: recursive numerical approaches to estimating parameters of the system model
Current limitations of this research: Quantitative, model-based in that the model structure needs to be given; has not been explored in application to systems in which hostile and deceptive agents are present.

Research field: Plan and Intent Recognition research
Potential contributions of this research to the Assessment Problem: approaches to recognition of enemy plans / intent
Current limitations of this research: current work has focused on non-hostile agents; little work has been done with respect to hostile, deceiving agent's plan and intent

Research field: Decision Theory, e.g., [4], [6].
Potential contributions of this research to the Assessment Problem: determining value of information
Current limitations of this research: real-time aspects are not addressed. While DT literature deals rather explicitly with certain temporal aspects, such as the time or stage of the decision process at which the information will be received, it doesn't consider deadline-type constraints.

Research field: Contingency Planning, e.g., [12], [13], [14], [15].
Potential contributions of this research to the Assessment Problem: understanding the expected contingencies and the possible responses to the contingencies provides basis for deciding what is worth observing
Current limitations of this research: Computing the expected impact on the plan of all contingencies is computationally very expensive; known techniques will probably not scale up to full-fledged military scenarios. Also, the real-time issues have not yet been addressed in this work. In this connection, we also point out an emerging research direction: integration or middle ground between AI planning and MDP planning. These two views are on the opposite ends of the spectrum of approaches to the Assessment Problem. The former views alternative actions (and required observations) as an outcome of the contingency planning process; the latter uses abstraction / aggregation / envelope techniques to define every action (and required observations) as a function of the set of states. We join a number of researchers in both of the two communities who believe that there is an opportunity for productive synergy of these two seemingly opposing views [6].

Research field: Bayesian and other Probabilistic Reasoning; e.g., [17].
Potential contributions of this research to the Assessment Problem: techniques for combining information elements and their values while accounting for their respective uncertainty
Current limitations of this research: strong model-dependence; limited study of applicability to adversarial problems

Research field: Statistical Signal Processing
Potential contributions of this research to the Assessment Problem: identifying and classifying patterns in information time series that are "normal" vs. "abnormal" (which might include, for example, signals that are "too clear" and might indicate enemy's deception) using statistical signatures
Current limitations of this research: confidence measures require models and prior distributions based on large data sets

Research field: Hypothesis Testing, Belief Revision
Potential contributions of this research to the Assessment Problem: confirming or disconfirming a hypothesis given the available observations
Current limitations of this research: highly model-dependent and not always computationally feasible for very large datasets. Also, real-time issues not addressed.

Research field: Anytime algorithms, e.g., [18], [19], [20].
Potential contributions of this research to the Assessment Problem: Research in this field addresses the problem of trading off the time of computation against the quality of the results produced. An anytime algorithm is one whose output quality increases monotonically with the amount of

time it is allotted; anytime algorithms can be interrupted at anytime and can produce a reasonable solution (hence their name). Consequently, with an anytime algorithm one can perform accurate sensing and extensive reasoning when time permits, and more coarse-grained sensing and limited reasoning when there are severe time pressures. This work directly addresses the real-time issues that are overlooked in so many of the other approaches mentioned.

Current limitations of this work: The construction of effective anytime algorithms remains very difficult, and their applicability to very large systems has not yet been demonstrated. One idea that has been explored is to decompose medium- and large-scale systems into collections of anytime components working together, but the technology is still limited for composing anytime algorithms in way that ensures anytime properties for the larger system.

7. Overarching and Emerging Research Issues

It does not appear that any of the research fields mentioned above has developed comprehensive techniques that apply to the military Assessment Problem with its inherent complexity, particularly its dynamically changing structure, real-time constraints and impacts of intelligent opponent.

To illustrate this point with an example, let us again take a look at the work on Model-Based Diagnosis. This research has made fairly significant progress: significant enough that it's being used in Deep Space One, which does involve a dynamically changing situation and real-time constraints. However, getting MBD to work for DS1 involves a very large amount of engineering; moreover the designers of the MBD component were also working closely with the systems engineers, so that they fully understood the performance of the system, and to some extent could control it. In a warfare situation, one side may fail to have a complete understanding of the enemy behavior, and they certainly don't have much control over it in advance. Therefore, questions remain about the applicability of the MDB approach of DS1. In fact, there are questions about its applicability even to other systems that are more similar to DS1 [2].

With a few notable exceptions most of the research fields relevant to the Assessment Problem assume the existence of a model of the phenomena or system in question. We use the term "model-dependent" to describe such approaches. Most of the fields of approaches we identified are strongly model-dependent in the sense that the quality of their solutions is entirely dependent on the quality and completeness of the available model(s). We question, however, the availability and durable applicability of models in the military domain. Although certain aspects of weapons and forces have been extensively modeled, comprehensive models of Battlespace and Command do not exist. One might argue that construction of comprehensive, reliable, validated and useable models of such complex systems and attending phenomena can be extremely expensive and lengthy effort. Perhaps even more importantly, one may question if any model of military matters, which by necessity are biased toward "the last war," will ever be applicable to "the next war." The models constructed with so much effort and expense may be hopelessly outdated by the time they are completed and ready for use. Thus, it appears that the practical approaches to solving the Assessment Problem must be able to perform effectively with at best a partially modeled world.

With this issue of model-dependency in mind, we offer that the Assessment Problem points to several emerging research issues: the need for model-independent approaches, or for means to construct models automatically and dynamically:

1. There is a need for synthesis of, or middle ground between the model-based and model-independent approaches, e.g., ability to enrich or to correct an existing partial model using a model-independent approach such as Machine Learning or Data Mining.
2. There is a need for approaches to automated or semi-automated construction of models, rapidly and dynamically during the execution of actions within the Battlespace. This need may be partially addressed by an effective methodology for disciplined construction of models; may include tools to support human modelers in very rapid construction of models from pre-existing flexible components. The search for such approaches might benefit from leveraging somewhat analogous ongoing research in techniques and tools to support rapid construction of knowledge bases.

Acknowledgments

Col McCorry, the Program Manager of the DARPA's JFACC program has provided the leadership required to initiate this study. The significance of the Assessment Problem has been recognized largely due to the arguments advanced by Mr. Brian Sharkey of DARPA. Bonnie McDonough and Mark Schwartz of Logica, Inc. have supported the brainstorming sessions of the authors and contributed important comments.

References

[1] R. Davis and W. Hamscher, "Model-based Reasoning: Troubleshooting," in H E. Shrobe, ed., *Exploring Artificial Intelligence: Survey Talks from the National Conferences on*

Artificial Intelligence, San Mateo, CA: Morgan Kaufmann, pp. 297-346, 1988.

[2] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams, "Remote Agent: To Boldly Go Where No AI System has Gone Before," *Artificial Intelligence* 103:5-47, 1998.

[3] R. L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, New York: Cambridge Univ. Press, 1993 (first published in 1976).

[4] P. Haddawy, A. Doan, and R. Goodwin, "Efficient Decision-Theoretic Planning: Techniques and Empirical Analysis," *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1995.

[5] P. Haddawy, A. Doan, and C. E. Kahn Jr., "Decision-theoretic Refinement Planning in Medical Decision Making: Management of Acute Deep Venous Thrombosis," *Medical Decision Making* 16(4):315-325, 1996.

[6] C. Boutilier, T. Dean, and S. Hanks, "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage," forthcoming in *Journal of Artificial Intelligence Research*, 1999.

[7] M. P. Georgeff and F. F. Ingrand, "Decision-making in an Embedded Reasoning System," *Proceedings of the 11th International Joint Conference on Artificial Intelligence (ICJAI)*, 1989.

[8] R. Doyle, D. Atkinson, and R. Doshi, "Generating Perception Requests and Expectations to Verify the Execution of Plans," *Jet Propulsion Laboratory Technical Report JPD D-3394*, 1996

[9] J. L. Fernandez and R. G. Simmons, "Robust Execution Monitoring for Navigation Plans," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotic Systems*, 1998.

[10] M. M. Veloso, M. E. Pollack, and M. T. Cox, "Rationale-Based Monitoring for Planning in Dynamic Environments, *Proceedings of the 4th International Conference on AI Planning Systems (AIPS)*, 1998

[11] C. A. Knoblock, "Building a Planner for Information Gathering: A Report from the Trenches" *Proceedings of the 3rd International Conference on AI Planning Systems*, 1996.

[12] L. Pryor and G. Collins, "Planning for Contingencies: A Decision-Based Approach," *Journal of Artificial Intelligence Research*, 4:287-339, 1996.

[13] D. Draper, S. Hanks, and D. Weld, "Probabilistic Planning with Information Gathering and Contingent Execution," *Proceedings of the 2nd International Conference on AI Planning Systems (AIPS)*, 1994.

[14] J. Blythe and M. Veloso, "Analogical Replay for Efficient Conditional Planning," *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)*, 1997.

[15] N. Onder and M. E. Pollack, "Conditional, Probabilistic Planning: A Unifying Algorithm and Effective Search Control Mechanisms," to appear in the *15th National Conference on Artificial Intelligence (AAAI)*, 1998.

[16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, CA: Morgan Kaufmann, 1988

[17] F. V. Jensen, *An Introduction to Bayesian Networks*, New York: Springer Verlag, 1996.

[18] T. Dean and M. Boddy, "An Analysis of Time-Dependent Planning," *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI)*, 1988.

[19] S. Zilberstein, "Using Anytime Algorithms in Intelligent Systems," *AI Magazine*, 17(3):73-83, 1996.

[20] S. J. Russell and E. H. Wefald, *Do the Right Thing: Studies in Limited Rationality*, Cambridge, MA: MIT Press, 1991.

[21] T. M . Mitchell, *Machine Learning*, New York: McGraw-Hill, 1997.

[22] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A Survey," *Journal of Artificial Intelligence Research* 4:237-185, 1996.

[23] D. Heckerman, D. Geiger, and D. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Machine Learning* 20, 1995.

[24] Aoki, M. "Control of large-scale dynamic systems by aggregation," IEEE Trans. on Automatic Control, pp. 246-253, 1968.

[25] Ramaswamy, G.N.; Verghese, G.C.; Rouco, L.; Vialas, C.; DeMarco, C.L., "Synchrony, aggregation, and multi-area eigenanalysis," IEEE Transactions on Power Systems, vol.10, no.4, p. 1986-93

[26] Chow, J.H.; Galarza, R.; Accari, P.; Price, W.W., "Inertial and slow coherency aggregation algorithms for power system dynamic model reduction," IEEE Transactions on Power Systems, vol.10, no.2, p. 680-5

[27] Miah, A.M., "Simple dynamic equivalent for fast online transient stability assessment," IEEE Proceedings-Generation, Transmission and Distribution, vol.145, no.1, p. 49-55

[28] G. Dullerud and R. Smith, "Sampled-data model validation: an algorithm and experimental application," International Journal of Robust & Nonlinear Control, vol. 6, 1996.

SimQL: an Interface for Integrating Access to Simulations into Information Systems

Gio Wiederhold, Hector Garcia-Molina
Stanford University, Computer Science Department.
Gates Computer Science Building 4A
Stanford CA 94305-9040
<gio, hector@cs.stanford.edu>

Abstract

We have performed proof-of-concept research on prediction software interfaces for information system. Such software can aid in the projection of effects of Command and Control (C2) decisions and allows alternate Courses-of-Action (CoAs) to be evaluated. Our objective is to significantly augment Command and Control decision services by making predictive results of simulations and similar software as accessible as databases and other information components. The motivating concept is that an interface language allows separation of customers and providers, and that the autonomy created allows progress to be made independently.

Motivation

C4I (C2, Communication, Computing, and Information) systems have grown in scope to include a variety of logistics, intelligence, tactical, and geographical databases, as well as message links, providing essential background for C2. However, the military commander primarily has to plan and schedule actions beyond the current point-in-time. Databases can make past and near-current data available, but cannot predict the future.

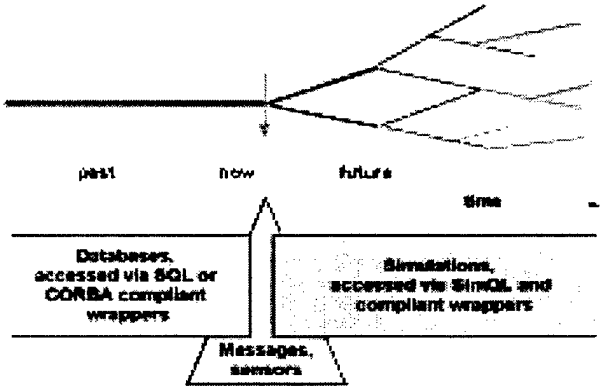
Diverse predictive tools come into play for projecting the effect in the future of decisions to be made now. These tools range from spreadsheets to war-gaming simulations. Major simulation programs are very costly and most are impossible to reuse in new settings [Zyda:97]. The most common tools used in practice is the spreadsheet, although its capabilities tend to be limited. They provide information which is complementary to the information about the past provided by databases, and help in selecting the best course-of-action [LindenG:92]. Quoting from "New World Vistas, Air and Space Power for the 21st Century" [McCall:96]: The two 'Capabilities Requiring Military Investment in Information Technology' are:

1. Highly robust real-time software modules for data fusion and integration;
2. Integration of simulation software into military information systems.

Today rapid progress is being made in information fusion from heterogeneous resources such as databases, text, and semi-structured information bases [WiederholdG:97]. Much of this work is ready for transfer to practical settings. However, the results of predictive tools have been rarely addressed in terms of integration and fusion [Orsborn:94]. Because of this gap, for most C2 decisions the capabilities envisioned by C4I advocates remain disjoint. When predictive tools are hard to use the commanders will rely more on their intuition than on computing tools.

Our vision for true C4I systems is sketched in Figure 1. Note that there will be multiple potential futures, leading to requirements for use of SimQL that are more demanding than we encounter today. But new system concepts also provide an opportunity to integrate technology, that is now dispersed and hard to use, into a new model.

Figure 1: Model for needed C4I capabilities



Past Work

We were funded by DARPA DSO for a small investigation to define and demonstrate a simulation access language, `SimQL'. Such a language is NOT intended for writing new simulations, but for providing information systems access to the results of existing predictive tools, via a wrapper infrastructure. The tools we explored ranged from simple spreadsheets, to weather forecasting, to computational assessments of future resource availability. Within the limited demonstration, we were unable to access fully distributed simulations as performed in military

training exercises (SIMNET [MillerT:95]), although we used such data in a related project [MalufWLP:97].

Concept

Technology has made great strides in accessing past information, stored in databases, object-bases, or the World-Wide Web. Access to information about current events is also improving dramatically. We now must expand the scope to include a peek into the future. Decision-making in planning, both in military and business environments, depends on knowing past, present, and future situations. For the latter we must access simulations. Many simulations are available from remote sites [FishwickH:98], they may also communicate with each other, as SimNet [Singhal:96], but are rarely accessible to be part of general information systems, so that we should handle both local and remote simulation services.

The concept of our simulation access language, SimQL, mirrors that of SQL for databases. Modern versions of SQL provide now also remote access [DateD:93]. Our projected ability to access simulations as part of an information system adds a significant new capability, by allowing simultaneous and seamless access to factual data and projections (e.g., logistics data with future deployment projections).

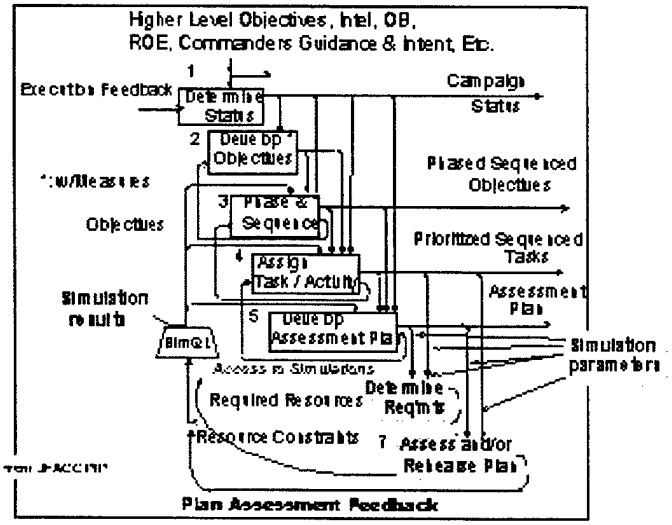


Figure 2: Simulation tool potential in JFACC

To enable rapid insertion of predictive services our SimQL interfaces can be built using emerging standard conventions for information systems. For instance, they might use a CORBA communication framework, and Java for client-based services. Such use of COTS technology will facilitate the integration of our SimQL interface to other systems that provide access to diverse non-predictive data resources.

To make the results obtained from a simulation clear and useful for the decision maker the interface should also use a simple model. Computer screens today focus on providing a desktop image, with cut and paste capability, while relational databases use tables to present their contents, and spreadsheets use a matrix.. However the objects to be described have a time dimension and also an uncertainty associated with them. We hence used a simple object model as the descriptive interface for SimQL.

Predictive Tools

To project the outcome of current decisions into the future, as every planner must, requires some form of simulation. In earlier generations, that simulation was done in the planner's mind, the planner would sketch reasonable scenarios and develop mentally alternate courses-of-action, focusing on those that had been worked out in earlier exercises. When the plans become complex, tools are needed for pruning, presentation, and assessment. Sand tables are still used for the training of military planners, but are increasingly being replaced by computer-based simulations. In business, spreadsheets are used frequently. Simple heuristics help the planners in pruning. For instance, military doctrine demands air superiority and that the forces and their armaments would be sufficient to overwhelm the enemy, while there would be enough uncommitted forces and logistical support to provide a reserve backup. While these rules still hold, the world is getting more complex, the resources are fewer, and more specialized, so that traditional heuristics are often barely relevant.

Rapid, ad hoc, access to information for planning, as exemplified by the use of DART during the preparations for Desert Storm, was extremely valuable, and superseded effectively many ponderous support planning systems that had been developed using older technology. However, DART could not execute arbitrary planning scenarios, and only one planning tool (originally developed for airport gate allocation by American Airlines) was adapted for use. Today, we have crucial simulations in all aspects of military and commercial planning. Logistics plans are developed by simulating alternate transport modalities, capacities, and risks. Production planners execute simulations to see how they can best exploit their resources. Financial planners use spreadsheets to work out alternate budgets. Most importantly, tactical plans are fully or partially simulated in all military exercises. The expectation, as cited in [McCall:96], is that this technology will transfer into military operations in the future. Even limited situational assessment of current status requires projection. Since C2 data are often out-of-date, commanders must routinely make undocumented projections even to judge the current readiness situation and obtain a complete tactical picture.

Lacking in today's practice is the ability to interoperate with even simple simulations at a direct functional level. Although interchange standards have been developed for the exchange of data objects within the SimNET initiative operated by the Defense Modeling and Simulation Office (DMSO), there is today no direct external access to its results, and no capability for its interoperation with existing military data and information systems. Access to predictive

tools through SimQL will provide C2 applications with a powerful, generalized interface to their results.

Specifics

The prediction tools we focus on are pre-existing, and will be wrapped to provide robust access to the model parameters and execution results. Wrapping provides a compatible 'machine-friendly' interface to exchange model information and respond to queries. We followed the model of SQL, which is not a language in which to write a database system - those may be written in C or Ada - but rather a language to select results for further use in information systems. Just as SQL provides access to a wealth of database technology and to database services that are often maintained by others, we expect that SimQL will provide access to the growing portfolio of simulation technology and predictive services maintained by others, and break the bottleneck which is now experienced when predictions are to be a part of larger systems that support planning for military, industry, or business.

There are two aspects to SQL which SimQL models:

1. A Schema that describes to the invoking program, its programmers, and its customer the accessible content.
2. A Query-language, which provides the actual access when the information resource is being used by the customer.

There are also some differences, of course, and some of them will require further investigation to assure effectiveness and seamless interoperation.

1. Not all simulation information is described in the schema. Simulations are often controlled by hundreds of variables, and mapping all of them into a schema is inappropriate. Only those variables that are needed for querying results and for controlling the simulation are externally accessible. The rest will be maintained by the simulation manager. Defining the appropriate schema will require joint efforts of providers and users.
2. Predictions always incorporate uncertainty. Thus, measures of uncertainty are reported with the results. In the future information systems that integrate the results will take uncertainty explicitly into account, so that the decision-maker can weigh risks versus costs.
3. Interoperation with past information is required. SimQL must integrate past, present, and simulated information, providing a continuous view. Furthermore, it must indicate what information is valid when. This is especially important since most databases are not fully up-to-date. Extrapolating stale database values to the current time can be a simple, but very useful task for a SimQL-based service.
4. Multiple courses-of-action (CoAs) must be supported. Multiple candidate alternatives may be valid simultaneously in some future domains. Thus, systems that access both databases and predictive information must deal with multiple courses-of-action, as indicated in Figure 1.

Current State

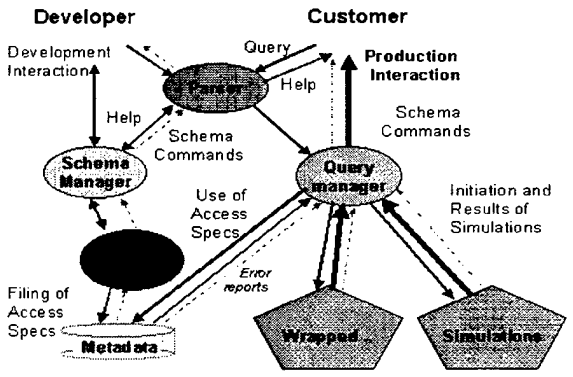
We have built a proof-of-concept implementation by modifying an existing public SQL implementation (RedBase). The benefit was to gain early on experience with compiling SimQL. The functions implemented were

- Parsing SimQL commands given by a customer
- Registering a wrapped simulation for a wrapper developer
- Creating simulation models for registered simulations and modifying those models arbitrarily (by the wrapper developer)
- Accessing a simulation through its model in SimQL and getting results back (by the customer)

The SimQL implementation includes four major component types. The three active portions are shown in Figure 3.

Figure 3: Components and information flow

1. The compiler for the SimQL language, which generates code for the schema and to query wrapped forecasting resources



2. A repository containing the schemas and metadata for the wrapped resources, identifying input and output parameters for each.
3. The actual forecasting tools, spreadsheets, discrete simulations, and web sources.
4. A wrapper generation tool to bring existing forecasting tools, as simulations, spreadsheets, and dynamic web sources into compliance

Specifics of the implementation are available on our web pages and in [Wiederhold:99]. Our experiments used diverse simulations. They were wrapped to provide information to a SimQL interface.

a. Two spreadsheets containing formulas that projected business costs and profits into the future. Inputs were investment amounts, and results were made available for years into the future.
b. A short-range weather forecast available from NOAA on the world-wide web. Temperature and precipitation results were available for major cities, with an indication of uncertainty, which rapidly increased beyond 5 days.

c. A long-range agricultural weather forecast for areas that overlapped with the cities. The initial uncertainty here was quite high, but increased little over a period of a several months.

d. A discrete simulation of the operation of a gasoline station, giving required refill schedules and profits.

A customer application can invoke multiple SimQL simulations. Our experiments only combined simulations b) and c), selecting the forecast based on data with minimal uncertainty over a wide range. Still, these experiments with a few real-world simulation convinced us of the applicability of the SimQL concept to a range of settings and provides a foundation for further research in this direction. Details of the prototype are given in [WiederholdJG:98] and on our webpages.

Conclusion

We have investigated the potential of a major augmentation for future C4I systems. Our results, indicate that diverse predictive tools may be accessed in an integrated fashion. However, much work lies ahead in order to support and implement the model of future information systems to support decision-making that we envisage.

An information model that supports multiple futures must provide facilities to combine uncertainties over its branches and estimate current or intermediate values from assessment of outcome values. Rapid recomputation is needed when situations change and after each time interval that forecloses alternatives that existed before. When the model of the future is extensive, pruning heuristics will be needed as well. Many related tasks have been investigated and demonstrated in isolated settings. Bringing all the tools to the decision maker remains an important challenge.

Acknowledgement

The original SQL compiler was written by Mark McAuliffe, of the University of Wisconsin-Madison, and modified at Stanford by Dallan Quass and Jan Jannink. Rushan Jiang adapted the compiler to SIMQL, created the wrapping wizard, wrapped several simulations, and created the interfaces. James Chiu, a Stanford CSD Master's student, provided and wrapped the gas station simulation.

The project was supported by DARPA DSO, Pradeep Khosla was the Program Manager; and awarded through NIST, Award 60NANB6D0038, managed by Ram Sriram.

References

Various reports on our current work are available at http://www-db.stanford.edu/LIC/SimQL.html.

[DateD:93] C.J. Date and Hugh Darwen: *A Guide to the SQL Standard, 3rd ed*: Addison Wesley, June 1993.

[FishwickH:98] Paul Fishwick and David Hill, eds: *1998 International Conference on Web-Based Modeling & Simulation*; Society for Computer Simulation, Jan 1998, http://www.cis.ufl.edu/~fishwick/webconf.html.

[LindenG:92] Ted Linden and D. Gaw 1992: "JIGSAW: Preference-directed, Co-operative Scheduling," *AAAI Spring Symposium: Practical Approaches to Scheduling and Planning*, March 1992

[MalufWLP:97] David A. Maluf, Gio Wiederhold, Ted Linden, and Priya Panchapa-gesan: "Mediation to Implement Feedback in Training"; *CrossTalk: Journal of Defense Software Engineering*, Software Technology Support Center, Department of Defense, August 1997.

[McCall:96] Gene McCall (editor): *New World Vistas, Air and Space Power for the 21st Century*; Air Force Scientific Advisory Board, April 1996, Information Technology volume, pp. 9.

[MillerT:95] Duncan C. Miller and Jack A. Thorpe: "SIMNET: The Advent of Computer Networking"; *Proceedings of the IEEE*, August 1995, Vol.83 No.8, pages 1116-1123.

[Orsborn:94] Kjell Orsborn: "Applying Next Generation Object-Oriented DBMS for Finite Element Analysis"; ADB conference, Vadstena, Sweden, in Litwin, Risch *Applications of Database'*, Lecture Notes In Computer Science, Springer, 1994.

[Singhal:96] Sandeep Singhal: Effective Remote Modeling in Large-Scale Distributed Interactive Simulation Environments; PhD Thesis, Stanford CSD, 1996.

[WiederholdG:97] Gio Wiederhold and Michael Genesereth: "The Conceptual Basis for Mediation Services"; *IEEE Expert, Intelligent Systems and their Applications*, Vol.12 No.5, Sep-Oct.1997.

[Wiederhold:99] Gio Wiederhold: "Information Systems that *Really* Support Decision-making"; in 11th International Symposium on Methodologies for Intelligent Systems (ISMIS), Warsaw Poland, June 1999, in Ras & Skowron Foundations for Intelligent Systems, Springer Lecture Notes on AI 1609, pp. 56-66.

[Zyda:97] Michael Zyda, chair: *Modeling and Simulation, Linking Entertainment and Defense*; Committee on Modeling and Simulation, National Academy Press, 1997.

A Market-Oriented Programming Approach to Advanced ISR Management

Dave Applin
SAIC
David.L.Applin@ saic.com

Dr. Paul McCoy
SAIC
Paul.F.McCoy@saic.com

Dr. Michael Wellman
University of Michigan
Wellman@umich.edu

Abstract

The DARPA Advanced Intelligence, Surveillance, and Reconnaissance (ISR) Management (AIM) project seeks to develop and transition technologies to tightly integrate the ISR management process with dynamic operational cycles. These technologies will provide capabilities for optimal requirements satisfaction and response to real-time, or ad hoc, information needs. One approach under investigation for the dynamic distributed allocation of ISR resources is "market-oriented programming."

1. Introduction

The phrase "market-oriented programming" refers to the general approach of deriving solutions to distributed dynamic resource allocation problems by computing the competitive equilibrium of an artificial economy. Within these economies, agents (producers and consumers) interact by offering to buy or sell quantities of commodities at unit prices which may vary over time. When the system reaches equilibrium, the computational market has computed a Pareto optimal allocation of resources throughout the system. The basic principles of market-oriented programming are:

- Commodities are priced through auction
- Commodities are exchanged based on the prices set
- Agents (both consumer and producers) have localized authority and information, and limited communication.
- Agent behavior is represented by computer code

2. Methodology

At a high level, we start with some number of goods and agents. Agents can be *consumers*, *producers*, or both. *Consumers* can buy, sell, and consume goods, and their preferences for consuming various combinations of bundles of goods are specified by their *utility function*. If an agent is a consumer, then its utility function ranks the various bundles of goods according to its preference. Consumers may also start with an initial allocation of some goods, termed their *endowment*. The objective of a consumer is to choose a feasible bundle of goods so as to maximize its utility function. A bundle is feasible for a consumer if its total cost at the prevailing prices does not exceed the value of its endowment at these prices.

Agents of the second type, *producers*, can transform quantities of goods of one type into quantities of goods of another type, according to their *technology*. The technology specifies the feasible combinations of inputs and outputs for the producer. This technology can be described by a *production function*, specifying the maximum output producible from the given inputs. The producer's objective is to choose a production plan that maximizes profits subject to its technology and the going price of its output and input goods.

The agents determine their demand at those prices by solving their corresponding constrained optimization problems and report the quantities demanded to the "auctioneer." Based on these reports, the auctioneer iteratively adjusts the prices up or down as there is an excess of demand or supply, respectively. The mechanization of the auction process can be through dealing with explicit bids to buy and sell single units, setting aggregate supply equal to aggregate demand, or using an optimization algorithm to establish prices and commodity exchanges. In the example described later we use linear programming.

When the prices stabilize we have what is called a competitive equilibrium as the set of prices and transfer of goods satisfy the following three conditions: (1) for these prices each consumer has maximized its utility subject to its endowment; (2) for these prices each producer has maximized its profit subject to its technology, and (3) the total amount consumed equals the total amount produced

plus initial endowments. As the framework for market-oriented programming is patterned directly after the general-equilibrium theory of economics, its welfare theorems apply. These theorems state that under assumptions (continuity, monotonicity, and concavity of the utility and production functions) competitive equilibrium exists, are unique, and are Pareto optimal (no agent can do better without some other doing worse).

3. Sensor Asset Management

In a simplified version of the sensor management problem, there are a set of available sensors, of various types defining their observation capabilities, effectiveness, operating costs, and other relevant attributes. There are also a number of observation tasks, of various priorities, defined by the types and amounts of sensors required to perform them successfully. In general, there may be many strategies to accomplish a given observation task. For example, multiple combinations of signal media (radar, optical, etc) and platforms (overhead, airborne, ground, etc) deployed at various locations for various times, may all provide some levels of observation quality. By selecting combinations (including configurations involving multiple sensor assets), market-oriented programming can achieve varying degrees of observation effectiveness.

To capture this situation in a model of producers and consumers, we can represent the primitive sensor assets by a simple kind of producer (with no explicit inputs) called *suppliers*. Since most assets are deployable over time to different activities, we allow the supplier to separately provide the asset in each time period. Setup and transportation times can be accounted for through intertemporal constraints on supply. The competing observation tasks are represented by consumers, who demand completion of these observations regardless of how they are accomplished. If there are various degrees of fidelity (or other quality attribute), these can be represented as distinct observation goods, with the consumer's preference expressing the relative value of the various qualities. In the middle, producers assemble configurations of sensor assets that can accomplish the observation tasks. Conceptually, there exists a distinct producer for every combination of assets over time that can provide a sensible level of observation task accomplishment. A sample sensor management task network schematic is in Figure 1.

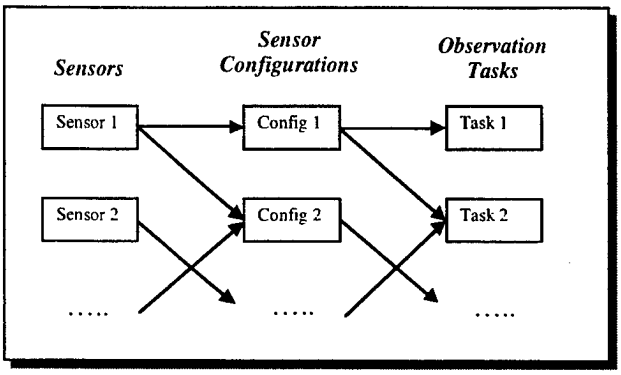


Figure 1. Sample Schematic Supply Chain for a Sensor Management Application

In some cases, it may simplify the description to decompose production into multiple levels, for example, one level may aggregate individual assets over multiple time periods (see Figure 2 for a fragment of such a network), while the next level aggregates multiple assets over time into an overall observation package.

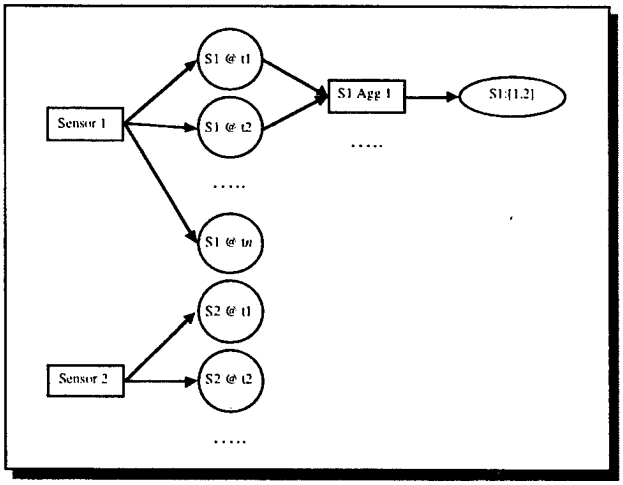


Figure 2. Temporal Aggregators

Of course, a realistic sensor application will have much additional structure specific to the context in which sensors are being deployed. Sensors may themselves require resources (e.g., fuel, computation, labor, communications), which would be modeled by making the sensing function a production activity requiring the sensor asset plus the enabling resources. The purpose of the observation may also be explicitly represented, thus capturing the value of sensing in terms of the uses toward which the observations are applied.

A resource shock in such a model could consist, for example, of lost sensor assets, lost enabling resources,

degraded sensor capability (e.g., that observation tasks require longer sensing periods than before). New observation tasks or sudden changes in relative preference for existing tasks would constitute a resource shock.

What sensors provide is essentially an information good. This leads to the view of a sensor network as an information services market, and opens up some interesting opportunities. For example, rather than directly exchanging sensor reports, agents could exchange *securities* tied to the uncertain propositions of interest.

A related idea is to consider the introduction of insurance markets. Insurance for uncertain events of consequence can lead to an improved allocation of risk, and provide incentives for proper monitoring of sensitive environment features. In the context of survivability, insurance can provide a way to explicitly hedge against adverse events, effectively arranging in advance for resource transfers more appropriate in case the specified contingencies obtain.

4. Experiment and Analysis

A detailed experiment was conducted to determine the feasibility of using a market-oriented programming approach for real-world scenarios. A Major Regional Conflict (MRC) (Korea) scenario was selected to stress the concept with large numbers of targets and ISR assets. The specific function of the market-oriented programming model was to allocate multiple platform and sensor combinations to imaging requirements over 5 x 24 hour intervals corresponding to the Air Tasking Order (ATO) cycle. The data used in the experiment is notional.

The SAIC-developed Reconnaissance Architecture Synthesis Model (RASM) was adapted to emulate an AIM MOP model as a starting point to speed prototyping. The AIM MOP model performed the functions of both producer and consumer agents as well as the auction mechanism for commodity auctions. The general inputs, processing, and outputs are shown in Figure 3 below. These parameters and processing are generally the same as those anticipated for the objective AIM configuration.

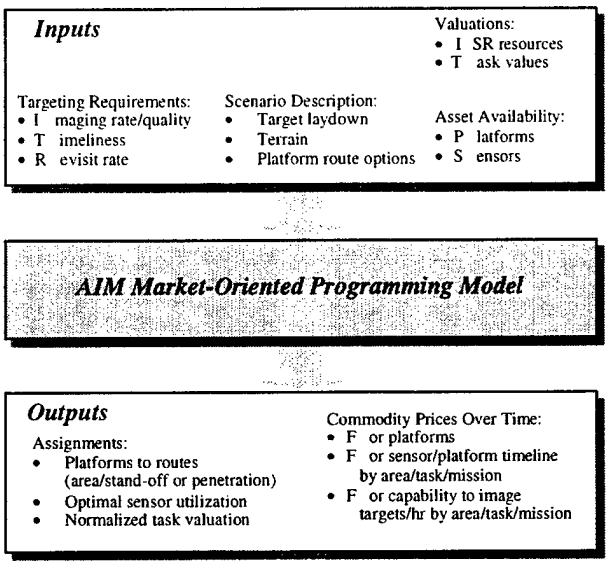


Figure 3. AIM Market-Oriented Programming Model

The experiment scenario encompassed a Major Regional Conflict in the Korean theater of operations. The first five days of a potential conflict were simulated. A notional target set was arrayed over DTED Level 1 terrain. Approximately 150,000 targets were represented in the baseline scenario and uniformly distributed within 15 scenario areas, or regions. Targets were of seven types: Land Target #1, Land Target #2, Air Target, Sea Target, Time Critical target (15 minute), Time Critical Target (2 minute), and Broad Area Coverage Target.

The scenario regions correspond to expected operational areas. These regions are shown in Figure 4 below and include Corps, Wing, Predator (Medium Altitude Endurance (MAE) UAV), Tactical UAV (TUAV), and Time Critical Target (TCT) regions. Each region has different target density, terrain and intervisibility, air defenses and platform survivability, and targeting requirements. Additionally, each region allowed standoff and penetration orbits for each platform, and view angles of North/South or East/West.

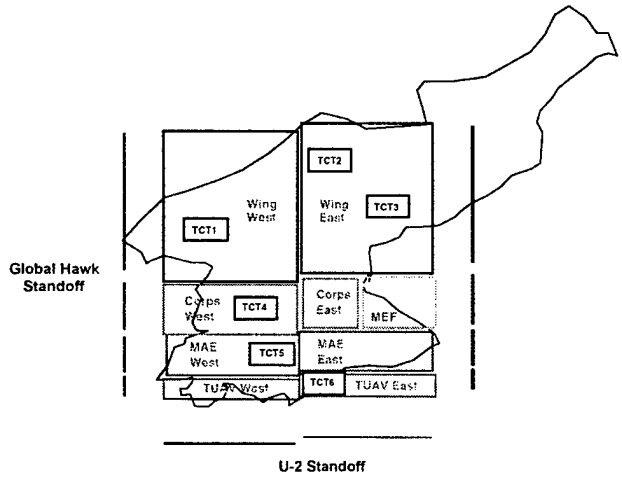


Figure 4. Scenario Graphic Model

Assumed ISR assets were four Global Hawk (spot and strip SAR), eight U-2 (spot and strip SAR), two Dark Star (spot and strip SAR), four Predator (EO/IR and strip SAR), and ten Tactical UAVs (TUAV) (EO/IR).

The experiment consisted of five runs of the AIM MOP model with each run representing a different day in the campaign. After the initial run, each subsequent run accounted for normal attrition of ISR assets and target "kills" resulting from the kill chain represented in the model.

The functions and supply chain of the AIM MOP model represented in the experiment are shown in Figure 5 and described below.

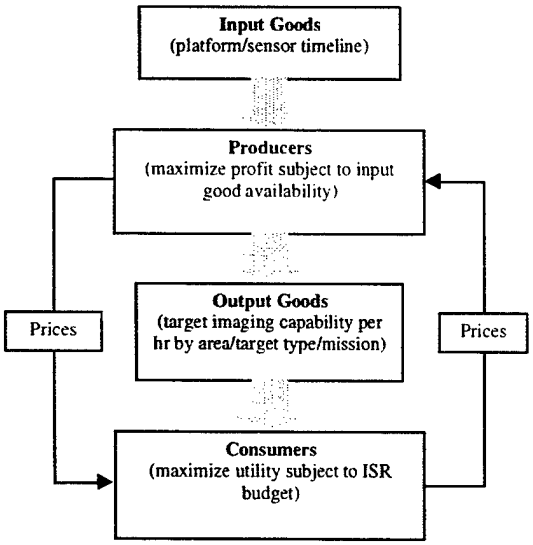


Figure 5. Commodity Inputs and Outputs

Input Goods

The input goods are ISR resource timeline (hours). The characteristics of these input goods are shown in the following table. Since the 14 platform and sensor combinations used in the experiment can be flown on 3 possible routes for each of the 15 scenario regions, the result is 630 platform/sensor/route input goods.

Table 1. Input Good Characteristics

Characteristics:

Platforms	Sensors
Speed	Type (spot SAR, strip SAR, MTI, EO/IR, ELINT)
Altitude	Range
Distance to base	Field of View (FOV)
Flight duration	Resolution/NIIRS
	Image formation time
	Probability of detection
	Restrictions on which sensors can be on at the same time

Output Goods

The output goods of the producers are the capability to image targets per hour. The characteristics of the output goods are: 1) quality of image, 2) timeliness, and 3) area coverage.

Producer Functions

Target Imaging Rate Calculations - The AIM MOP model computed the ability of each input good (platform, sensor, and route) to generate each output good (target imaging rate for each target type, area, and mission). This is potentially 630x420 combinations based on 630 platform, sensor, and route input goods (14 platform and sensor combinations that can be flown on three possible routes for each of the 15 regions), and 420 output goods (30 sub-areas (15 regions x two operations - stand-off or penetrate) x seven target types x two mission types (planning or execution)).

The rate calculations take into account the following:

- Maximum sensor imaging rate capability
- Platform speed and altitude
- Sensor Field of Regard/Field of View
- Sensor image quality
- Partial image quality satisfaction adjustment
- Image formation, processing, and exploitation time delay

- Target bonusing (where satisfying one requirement satisfies other requirements)
- Platform survivability
- Terrain intervisibilty
- Target density

Consumer Functions

The AIM MOP model represented the consumer functions as well. Consumers are the component commanders of the JTF whose objective is to maximize the value of the targets imaged.

Target Valuation - For the purpose of this experiment, target value was computed as the inverse of the revisit rate. This value is relative across all the target types. For example, the revisit rate for Corps West Time Critical Targets was every fifteen minutes. The corresponding target value for this type was four.

This initial scheme for valuing targets may not represent the true operational application. Commanders may choose to establish a uniform ranking for target types based on some other criteria, such as capability or perishability, that are then input to the MOP model as target values. Any scheme would have the same effect, which is to establish the consumer's objective function

Auction Mechanism

The auction mechanism in the AIM MOP model solves all consumer and producer optimization problems simultaneously in one linear program as prices and the interconnecting constraints are common to all problems (producer and consumer). The objective is to maximize the sum of the consumer (maximize utility) objectives. The producer (maximize net revenue) objectives are implicitly optimized. The constraints are all the constraints of the consumer and producer problems plus the interconnecting relationships. The linear program results in:

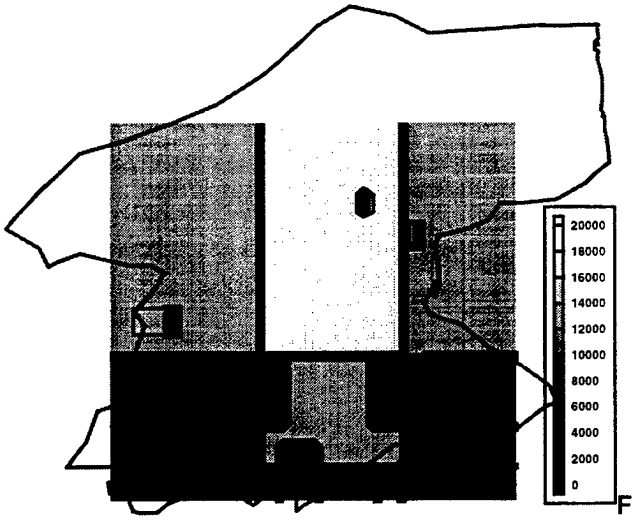
- Prices that are the optimal dual variables (assumes complete knowledge of all consumer and producer problems)
- Solved allocations that are Pareto Optimal at the solved prices
- Allocations that are globally optimal under the assumption that consumers have equal weight.

The linear program had approximately 2,000 constraints and 4,000 variables and was solved using MINOS.

Results

The results of the MOP auction mechanism given the inputs discussed earlier are described below. In general, the MOP model behaved in an expected and predictable manner with consistent results. Of particular note, the prices determined by the auction mechanism generally reflected the consumer values, except where exceptionally high costs associated with threat risk or scarcity of resources prevailed. In these cases, the resulting prices serve as effective feedback to consumers relative to their value of information and the cost of obtaining that information.

Figure 6 shows that prices are higher in the North as fewer platforms can access targets there. Prices are higher in the center at it requires penetration



igure 6. Output Commodity Prices on Day 1

As shown in the Figure 7 below the relative prices for Predator and Tactical UAV increase over time as they suffer the most losses due to attrition. The relative price for Dark Star increases as it is the only other platform that can penetrate hostile air space. Any changes in scenario conditions could cause price fluctuations (e.g., changes in requirements, asset availability, platform survivability, available routes, weather, etc.).

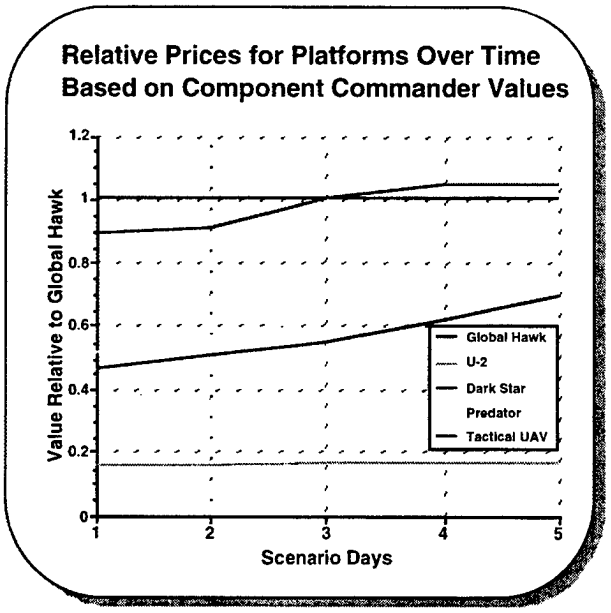


Figure 7. Platform Prices over Time

One measure of the performance of the AIM MOP model was the percentage of requirements satisfaction by component. The graphic below indicates that the JFACC and TCT target requirements are the most difficult to satisfy with the assumed inventory of sensor platforms. The percent of requirements satisfied is related to the number of assets that are available. It can be assumed that a higher level of requirement satisfaction would be possible if more assets were available.

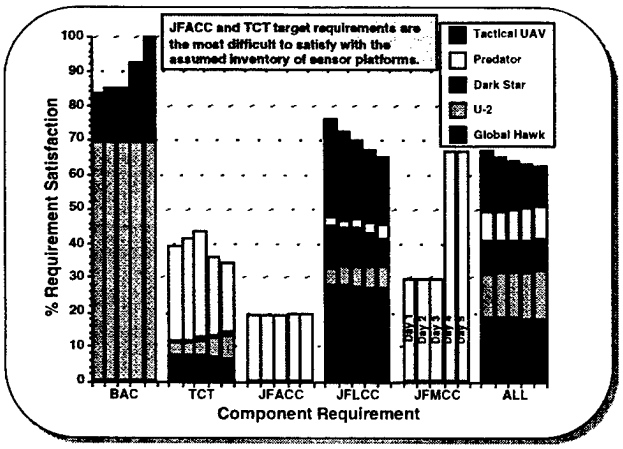


Figure 8. Requirements Satisfaction Breakdown

The following are summary conclusions from the initial experiment using the market-oriented programming model and sample MRC scenario:

- The market-oriented programming paradigm can be applied to the AIM problem of scheduling assets.
- An application of market-oriented programming can quickly generate a first-order allocation of platforms to routes and sensors to tasks.
- Prices of ISR assets give the JTF Commander insights as to critical resource needs.
- Prices of ISR tasks incorporate additional information beyond task values. This feature needs further examination.

These are additional observations that indicate possible areas of new or continued research:

- Prices can be used to give a first order response to changes. If an additional must-do task comes in, an immediate response could be to assign timeline from the sensor/platform with the lowest price and taking away timeline from those tasks with the lowest output prices. A similar dynamic reallocation could be performed in the face of sudden loss or degradation of assets, changes in priorities, or other external events.
- Use of commercial ISR products can be easily accommodated by representing them as a separate producer. Inputs would be real dollars (e.g., limited by ISR commercial budget of the JTF commander). An interesting derived output would be price (in terms of ISR consumer value) of each real dollar spent on commercial ISR products.
- The situation of two AIMs in separate theaters competing for overhead ISR support can be accommodated in the MOP paradigm by having the overhead asset manager allocate resources based on bids from both theaters.

5. Application to Advanced ISR Management

The AIM problem is characterized by multiple diverse and distributed information resources (producers) and equally diverse requestors of information (consumers) who have varying values for their information needs. The challenge for ISR management is to dynamically allocate information resources to meet consumer needs while attaining the highest total information value over a given problem set. Market-oriented programming addresses this challenge by accomplishing the following functions:

- Establishing a common scale for the importance of collecting information about different types of targets, threat missions, and areas of interest as a function of operational context, resource utilization, time and data quality.
- Providing a metric for the military utility of ISR objectives.
- Providing a common means of communicating value across DoD and commercial information resources.

- Providing continuous feedback to producers and consumers on information need satisfaction and operational conditions (e.g. supply and demand of resources).

Applying the MOP approach to ISR management involves constructing computational agents for ISR consumers and ISR producers, and constructing a framework (market) in which these agents can interact. The ISR consumers can be military commanders, such as the Joint Forces Commander (JFC) and subordinate component commanders. The computational agents representing these consumers will generate offers to buy ISR task satisfaction reflecting the values of the organizations they represent. Likewise, producer agents representing the ISR asset managers must also place a value on their resources. The producer agents will function to determine the types and quantities of ISR assets that could satisfy the consumers' tasks and keep track of their asset's availability and be motivated to maximize the accumulation of ISR currency. This distribution of goods and services constitutes the market solution to the ISR planning problem.

The commodities traded are ISR asset timeline (e.g., timeline on a U-2 sensor system, etc.). The currency for these transactions can be information tokens or some other currency (real or fictitious). An example AIM scenario using MOP may be that a single commander, say the JFC, gives his subordinate commanders information tokens each can spend to perform ISR tasks according to their own set of values. Each consumer agent would start with a list of ISR tasks (including what/where/when/etc.) and then generate a bid for satisfaction of each task. One or more producer agents would review these bids and generate an offer to satisfy them. A key question is evaluating the desirability of having multiple separate producer agents (e.g. HUMINT, IMINT, SIGINT, RADINT) all bidding separately for tasks. A market framework provided by the AIM application software would clear the transaction using its rules and perceived market conditions and would notify the winning bidder, if any.

6. Summary

Market-oriented programming provides a principled approach to decentralized resource allocation. The general methodology is widely accepted as evidenced by the explosion in on-line auction sites currently used as a medium to exchange goods and services between distributed producers and consumers.

In the context of advanced ISR management, market-oriented programming potentially enables several key functions. These are:

- Integration of operational and intelligence planning
- Optimal multi-asset synchronization
- Dynamic prioritization and valuation of information needs
- Distributed collaborative planning
- Adaptive and extensible resource allocation environment

Because of its general applicability, market-oriented programming can also be used across a number of other economies, such as logistics, air operations, and fire support that are also characterized dynamic decentralized resource allocation.

Knowledge-Based Automatic Verification and Validation for Business Models

Yun-Heh Chen-Burger, David Robertson
Artificial Intelligence, Division of Informatics
The University of Edinburgh
80 South Bridge, Edinburgh, UK
emails: jessicac@dai.ed.ac.uk, dr@dai.ed.ac.uk
Tel:+44 131 650-2713, Fax:+44 131 650-3090

Jussi Stader
AIAI, The University of Edinburgh
80 South Bridge, Edinburgh, UK
email: jussi.stader@aiai.ed.ac.uk
Tel:+44 131 650-2732 Fax:+44 131 650-6513

Abstract

The potentially great rewards offered by applying business modelling methods to achieve greater understanding of the enterprise have convinced many businesses to use them. Unfortunately, there is a problem with these methods: it is difficult to determine and assure the quality of the built model, a problem which is partly rooted in the fact that most of them are informal. To address this problem we use first order predicate logic to capture the knowledge of a business modelling method as well as the models themselves. This provides the necessary computational platform for automatic verification (of modelling techniques) and validation (of the business models) support during the development lifecycle of the model. The role of the formal notation in this case is not to provide a formal semantics for the given method, but to provide a framework for sharing the information supplied at different modelling stages and for automated analysis of the model using logic and case-based reasoning techniques.

Key-words Business Modelling, BSDM, Organisational and Enterprise Modelling, Formal Method, Independent Verification and Validation.

1 Introduction

A variety of modelling methods have emerged that provide structure and notation to capture and analyse business strategies and practices of an organisation: for example, methods for business modelling (BSDM [6], Handbook of Organisational Processes [9]), process modelling (IDEF3 [10], PIF [8], PSL [12]), enterprise modelling (IDEF0 [11]), business process re-engineering (Hammer [5]) and business specifications using ontologies (Uschold [13]). Object-oriented technologies are also used to describe businesses and their activities: e.g. *Business Models* offered by *Rational Rose* [4], making use of *Activity Diagrams* in UML [1] and methods by Jacobson [7]. By providing a structural framework, these methods help an enterprise to capture its enterprise-wide knowledge which forms the basis for targeted analysis and helps the re-designing and re-shaping of a business. They also provide a neutral forum where people of different disciplines can communicate with each other. The purpose of these methods is to seek ways to improve an organisation's effectiveness, efficiency and profitability. The potential benefits can be tremendous for an enterprise, though not all applications of these methods have been equally successful.

A key factor when using such methods is to ensure that the produced model is the right one for the organisation, i.e. that it is an accurate representation

of the real world. However, it is hard to determine the quality of the produced models. Several typical problems are given below.

- *Lack of comprehensive understanding of an organisation:* A modern enterprise today is often a virtual entity which consists of many sub-organisations spread across many different geographical areas each with special functionalities and business characteristics. It is difficult to gain a complete and integral view of such an organisation. Furthermore, business goals and activities may change as an organisation has to react to today's fast moving global economic developments.

- *Gap between methodical and managerial knowledge:* To judge the correctness and appropriateness of a built model, it requires both expertise of the applied method and knowledge about how the method can be used appropriately for an organisation. This involves both modelling capabilities as well as management level knowledge of the organisation, and few people possess both.

- *Time pressure:* Most modelling projects are subject to a very limited time frame, with the result that once the model has been built, the modellers are often left with little or no time to sufficiently verify and validate their model, a task which, if it has to be carried out by hand, can be time-consuming.

- *Informal and semi-formal descriptions:* To allow the maximum flexibility and capability in describing a complex business environment, informal and semi-formal descriptions are used in parts of many of these methods. This, however, adds to the difficulties: the inherited ambiguities in natural language give rise to possible misinterpretations of models.

- *Lack of efficient and effective communication means:* A modelling method is intended to provide a communication forum for enterprise knowledge transfer between managers and software system developers. However, most methods do not have wide usage at this stage and would require additional training for staff. Furthermore, the description of a complete enterprise model can be too complicated for unaided human comprehension.

- *Model dynamics are implicit and complex:* The system dynamics that are implied or prescribed in the static descriptions of these methods are in general very complicated. Hence, in addition to the static aspects of the model, the dynamic properties of it may also become too complex for un-aided human reasoning.

To address these problems, we report on work that is aimed at providing automatic model building support based on a logical formal method. This paper focuses on the automatic verification and validation of such business models within the *Plan-Build-Verify-Validate-Refine* development lifecycle and the benefits that can be obtained from this: *verification* is the process of checking that no model rules or guidelines have been violated, *validation* is the process of confirming that the model is a true representation of the described world. The method used in this study is IBM's Business System Development Method (BSDM)[6].

2 Business Modelling in BSDM and its Extensions

The first activity in BSDM is *business modelling* which is an informal enterprise modelling method that captures the components of complex business environments and the rules which govern them. A business model serves as a communication medium between modellers and captures the converged view of senior managers of the company.

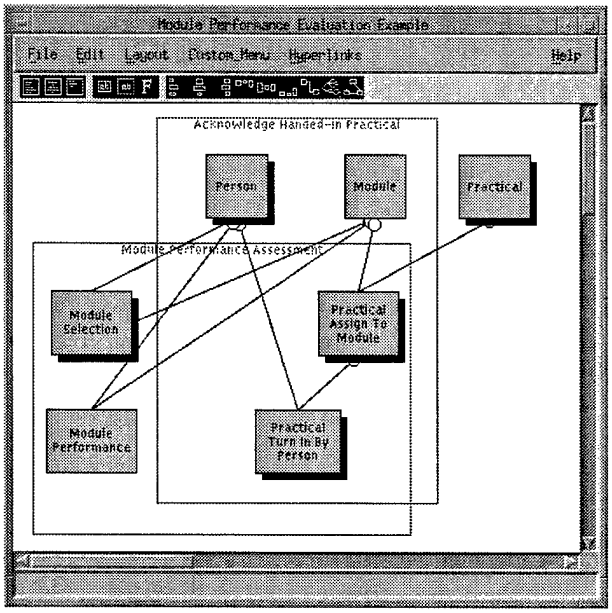


Figure 1: **An example BSDM business model**

Figure 1 shows a simplified BSDM business

model. The main components of a BSDM business model are entities, dependencies and processes, described in a graphical notation, supplemented with attributes and business rules in textual forms. Initially, an *entity model* is built which captures the concepts (abstract and physical things) in the business as *entities* (shown as rectangles) and the relationships between them as *dependencies* (shown as links between them). The entity attributes and business rules are captured in *entity definition forms.* The entity model is extended with information about the *processes* (shown as the greater rectangles which surround entities) which manage these entities to form a *process model.* A BSDM process describes the context of a business process, the circumstances which trigger such a process and the effects of its actions. Process attributes and rules are recorded in *process definition forms.* The relationships between entities and processes are specified using *entity functions.* In parallel to the development of entity and process models, *life cycle diagrams* are built. They describe information about an entity's life statuses and how different processes enable the transition between these life statuses. They also indicate the subtle relationships between processes and the operations used to carried out a particular task.

When a process model is instantiated with entity and process instances (occurrences), it depicts the dynamic state of the model at a particular point of time. However, since the business processes are described at a high level of abstraction, the information needed to capture system dynamics is absent in traditional BSDM models. In response, we introduce a new layer, the *procedural model*, on top of the entity and process model, which extends the process model and enables a declarative specification and automatic execution of business processes. This facilitates the explicit presentation as well as forecasting of dynamic aspects of a business model. This entity/process/procedural layered modelling approach is illustrated in more detail in [2].

3 Modelling Support Framework

Figure 2 gives a modelling overview for our developed system, *KBST-BM.* There are three phases: *modelling, verification* and *validation.* The tool provides various *GUIs* (Graphical User Interfaces) which allow the user to build *BSDM business models*, *life cycle diagrams* and *procedural models* and specify *user-defined rules.* The embedded BSDM

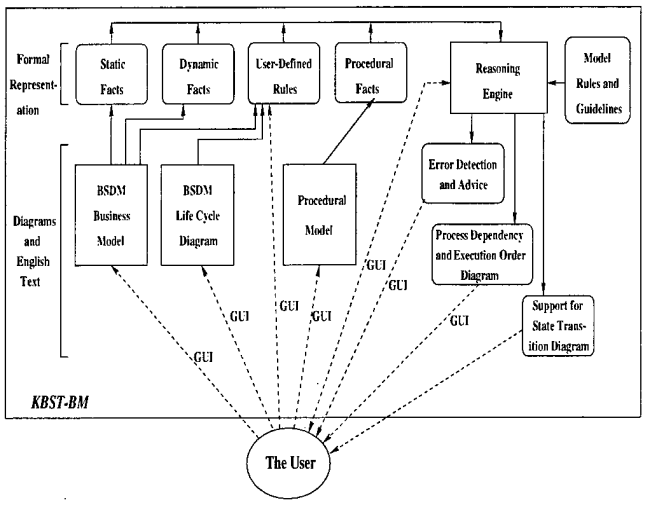


Figure 2: **The modelling support overview**

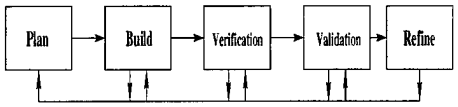


Figure 3: **Iterative development lifecycle**

knowledge, *model rules/guidelines*, together with the formalised input from the user are fed into the *KBST-BM Reasoning Engine* to provide automatic *verification* and *validation* facilities. This provides a framework for the modellers to carry out an iterative model development, i.e. to follow the *Plan-Build-Verify-Validate-Refine* lifecycle as shown in Figure 3.

At the beginning, business managers together with a BSDM facilitator create the initial business model. The emerging model, including the *life cycle diagrams*, is recorded in conventional BSDM notation using *KBST-BM.* From the graphical and textual description of the entity/process model, *KBST-BM* automatically derives a formal representation of the corresponding static and dynamic facts of the model. In addition, a set of user-defined rules are derived from the *life cycle diagrams*, through which the user specified entity life statuses and their transitions. Other rules which are explicitly given by the user are also formalised. General BSDM model rules and guidelines are considered to be fixed and, therefore, require no user input. They are already formalised and integrated as a basic component of *KBST-BM.*

4 Verification

The formal representation of the model as well as the rules provide input to the *reasoning engine* of KBST-BM which analyses the business model. An analysis report is produced which describes any violation of model rules and makes recommendations for how the model can be improved. The user can then modify the business model accordingly and start the next iteration of this process. The cycle is repeated until no further errors and recommendations are produced, or when the user decides not to incorporate any more of the given recommendations. Once no more model rule violations are reported by the reasoning engine, verification of the model is complete.

The verification support is based on recommendations extracted from the original BSDM method, which were formalised using *first order predicate logic (FOPL)* and implemented as *CLIPS* rules. We classify these recommendations into two categories, *model rules* and *model guidelines*, according to their strength of enforcement on models. *Model rules* are methodological BSDM rules and strongly recommended if the model is to be sound. *Model guidelines* reflect good modelling practice under normal circumstances but may be overridden by specific business requirements.

Model rules and guidelines are represented as axioms in our formal language. Two implication symbols are deployed: '$\Rightarrow$' is used to represent a model rule and is read as 'must be'; '$\triangleright$' is used for model guidelines and is read as 'should be'. A formula 'P $\Rightarrow$ ($\triangleright$) Q' reads 'if P is true then Q must (should) be true'. A simple model rule, which states that if an instance X has properties $p_1, p_2, ...p_n$, then the instance X must also have property Q, can be formalised as:

$$\forall X.p_1(X) \wedge p_2(X)... \wedge p_n(X) \Rightarrow q(X).$$

To provide automatic verification, two parts must be determined: the condition which identifies a violation of the model rule and the appropriate advice that should be given when the rule has been broken. To identify the violation, the given axiom is negated and normalised to describe the conditions when a model rule has been violated. The process of normalisation is to take away the implication symbol '$\Rightarrow$' and to transform the resulting formula to its simplest possible form, preferably using only '$\wedge$' and '$\neg$' logical connectors. A formula which includes complex constructs and multiple '$\vee$' connectors is translated into multiple distinct descriptions to cover all violation possibilities. The above axiom is negated and normalised below.

Negation:
$\neg(\forall X.p_1(X) \wedge p_2(X)... \wedge p_n(X) \Rightarrow q(X))$

Normalisation (1) :
$\exists X.\neg(\neg(p_1(X) \wedge p_2(X)... \wedge p_n(X)) \vee q(X))$

Normalisation (2) :
$\exists X.(p_1(X) \wedge p_2(X)... \wedge p_n(X)) \wedge \neg q(X))$

The final resulting formula states that a violation for the above axiom has occurred when "an instance X is found to have property $p_1, p_2, ...p_n$ but it does not have property q". Using this as a pre-condition statement and the formalised advice derivable from BSDM as a conclusion, the corresponding model rule can be written in *CLIPS* and *critiques* provided to the user as follows (where *model_rule_advice* is the function which generates advice for the user and $rule_n$ is the unique ID for the particular model rule):

CLIPS Model Rule N:

$(p_1(X) \wedge p_2(X)... \wedge p_n(X)) \wedge \neg q(X)$
$\Longrightarrow$
$model\_rule\_advice(rule_n, X)$

Based on the static and dynamic facts of the business model (which were automatically derived from the BSDM diagrams and are now stored in the system's knowledge base), the *CLIPS* inference engine can determine which modelling rules have their pre-conditions satisfied and then activate the corresponding function *model_rule_advice*, thus giving the user the appropriate feedback to his/her model. Similar formalising and normalising techniques are used for *model guidelines* but are communicated to the user simply as suggestions (using the function model_guideline_advice($guideline_n$, X)).

Model rules and *model guidelines* which target similar aspects of a model are grouped together to enable an iterative, systematic and topical verification process. This gives the user the freedom either to run a complete check on the model or to work on a particular aspect of the model, i.e. to use certain critiques only. This helps the user to focus on a particular design issue and not be overloaded by too much advice which is of no immediate concern.

Furthermore, each of these model rules and guidelines are implemented modularly and operate independently. This enables the user to verify as-

pects of the model in any preferred order. It further provides a flexible structure for possible future modifications or extensions to the rule base.

Given a (possibly partial) business model, several types of *critiques* and their advice for possible corrections are provided for the modellers, as listed below:

- *correctness* critiques detect syntactical and semantical errors;
- *completeness* critiques identify incomplete information in the model and suggest which missing concepts may need to be included;
- *consistency* critiques point out discrepancies existing across different parts of the model;
- *appropriateness* critiques show deviations from standard practices;
- *presentation* critiques highlight awkward use of naming style which can lead to misunderstandings or conceptual errors;
- *alternative* critiques search for similar models and present them as alternatives to a given modelling decision. This critique makes use of a case library of standard and past business models (we describe the use of case-based reasoning techniques in our *generic model advisor* in [3]).

5 Validation

Process Dependencies and Partial Execution Orders

To simulate and thereby validate the model, the user must create a *procedural model*, which can then be formalised and fed into the reasoning engine. However, before the *procedural model* is built, the user can gain an insight into the relationships between processes and an initial view of possible process behaviours through *process dependency and execution order diagrams*. These diagrams are automatically generated by the inference engine at the end of the verification phase without the need of any further user input. Although these dependencies are not explicitly captured in the original BSDM method, the partial ordering on process execution they describe can be derived from information which is already contained in the model, such as life cycle diagrams and entity function descriptions.

Figure 4 shows an example *process dependency and execution order diagram* for a business model

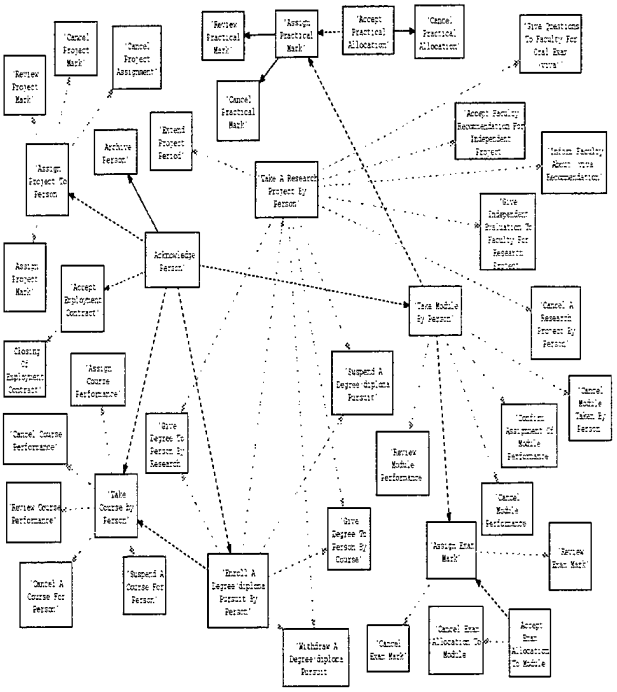


Figure 4: **Process dependency and partial execution order diagram**

which captures the context of module assessment for taught degree courses and for degree by research courses at a university. Forty-two processes have been described in the business model. Each is represented as a node in the diagram.

Four types of process dependencies are derived from a business model, each is represented by an arrow leaving from the process node which takes execution precedence to the dependent process node. Each process dependency type is represented by a different colour code in the diagram, which cannot be shown in the black-and-white figure in this paper.

Process dependency Type I dictates the strongest execution constraint and the most direct execution order relationship between two connected processes: the dependent process must only be executed after the preceding process has been executed, but can be executed immediately after the preceding process has completed. This dependency indicates that two processes work very closely together in a particular business operation.

Process dependency Types II and III each indicate a gradually weaker constraint and impose a less direct execution order. The weakest dependency, Type IV, indicates the least restrictions: the execution of the dependent process may or may not require the execution of the preceding process. In

the absence of the preceding process, the dependent process may decide to carry out (parts of) the functions which are normally performed by the preceding process to provide information that is needed for its own operations. In this case, the execution of the other process is bypassed.

Such automatically generated *process dependency and execution order diagrams* are beneficial to the user, since they allow better planning of what-if business scenarios by providing help in identifying suitable starting scenarios for simulation and by cutting down the search space of possible scenarios.

This model validation support is facilitated by the automatic analysis and reasoning abilities of *KBST-BM*. The underlying inference rules are also formalised using first order predicate logic and are implemented in *CLIPS*.

Business Model Simulation

BSDM's business processes give declarative descriptions of the "things" that are used in the process and the actions that are performed on these "things". BSDM does not capture details about how these actions are carried out or a deterministic order for carrying out these actions. In order to enable automatic simulation of processes in KBST-BM, however, some commitments to execution order are required.

The extended *procedural model* has been designed to allow the user to specify a *logical* execution sequence of processes which is independent of an organisation's current working practice and therefore can be implemented in different ways depending on the organisation's specific requirements. This fully complies with BSDM's business modelling principle, i.e. to provide a directorial view as opposed to an operational one, and therefore to provide a more stable view of an organisation.

Figure 5 shows an "uninstantiated" generic *procedural model* for a BSDM process. It specifies the structures and components of a procedural model but leaves their parameters uninstantiated. These parameters are details about the described process and will be shown in the empty brackets "()" when instantiated. The arrows show the flow of control between activities. Several types of standard components are specified in the layered procedural model. From top to bottom: the starting point of the process, process trigger, process preconditions, actions, another layer of actions, process postconditions and the ending point of the process. All layers of components are connected by an "and" connector which acts as a synchroniser to ensure all activities

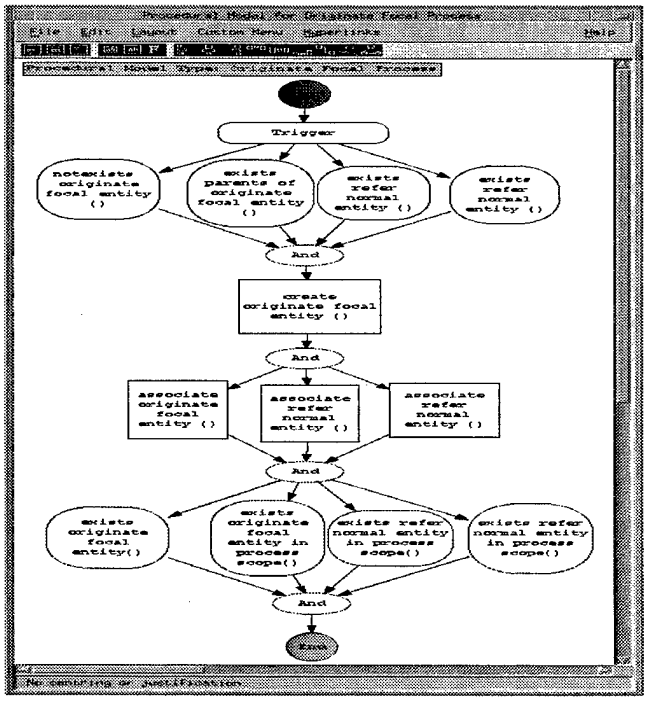


Figure 5: **A generic procedural model diagram**

of one layer have been finished before the next layer of activities is carried out. We provide a library of generic procedural models which the user can easily adapt for specific requirements.

After the user has constructed a procedural model for each process, facts about the dynamic states of the business model can be specified and the simulation can begin. During the simulation of the model the user interacts directly with the reasoning engine — telling it which process to execute from which starting point. The reasoning engine generates and maintains the state transition of the system and information about the various states, which the user can compare with the effects of business processes in the real world. The inference engine also detects and reports to the user any potential process execution conflicts. The user can decide to alter the simulation scenarios or to make necessary changes to the business model and then continue to validate it using the reasoning engine. If desirable, the user can also decide to "roll back" the current state to a previous state and explore alternative routes from there. The user can repeat the *plan-build-verify-validate-refine* model development cycle until the model has reached a satisfactory standard.

The *Business Model Simulator* is written in *Prolog* (taking advantage of Prolog's declarative programming style and its backtracking algorithm), uses the formal representation that has been auto-

matically derived from the business model by *KBST-BM* as input and generates model dynamics using the same formal method. The algorithm for the simulator is given below.

1. *If the current simulation time step is finished then stop the simulator and report the result of the simulation to the user; otherwise, go to 2.*

2. *Check all process triggers in the system. If all of the preconditions for any of the processes are satisfied and that the designated* **starting time** *(for execution) for that process is now or has passed, then put process in the* **Process Agenda**. *Go to 3.*

3. *Check all processes in the process agenda and collect those processes with an* **ending time** *which is now or has pasted in a set. Perform a* **process conflict check** *on all of the processes in the set; if there is any contradictions found between these processes then go to 4; otherwise go to 5.*

4. *Report to the user any contradictions found between processes, including detailed information about what has caused the problem and a brief suggestion for conflict resolution. The user can decide if any of the processes should be removed from the process agenda and thus from the system. After performing the requested removal operations, go to 3.*

5. *The user can now select those eligible processes to be executed. Each selected process is checked again to ensure that all of the process preconditions are still satisfied, and that the actions within each process is syntactically correct and the detailed requirements for executing each action are also satisfied. The user may ignore some (or all of the) processes for execution thereby exploring a specific execution route. The ignored processes are left unchanged in the process agenda. After each process execution, the postconditions are checked. Report to the user, if any irregularities have been found. Go to 6.*

6. *Advance the system to a new time. Advance the system to a new state if any changes have been made to the current state. Enquire the user whether he/she wishes to "rollback" the system and specifically which state he/she wishes to restore. If the answer is yes, restore the system to the specified state. Go to 1.*

The inference engine generates the dynamic behaviour of the model, i.e. it simulates the execution of processes, according to the instructions given by the user. The execution history is maintained in a *state transition diagram*, the details of which are represented by the underlying formalism.

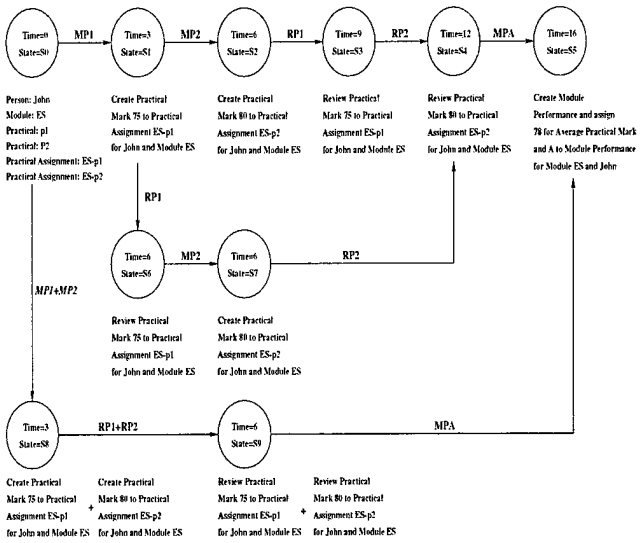


Figure 6: **An example state transition diagram**

Figure 6 shows an example *state transition diagram*. Each node represents the dynamic state of the model at a particular point of time, detailed comments about these states are given beneath these nodes, where each arrow denotes the process(es) which have been carried out to generate these system states.

Since multiple processes may be executed concurrently, several types of conflicts between processes are detected: the conflicting handling of data items, error handling of data items, and the prolonged waiting of processes in the *Process Agenda*.

Given both of the above validation facilities, the modellers can carry out the following tests on the process model:

- checking of the *appropriateness* of the operations described in a process context;
- checking of the *consistency* of the handling of the same (group of) entities by different processes;
- checking of the *redundancy* of operations that have been covered by more than one processes and *process merging* possibilities;
- checking of the *completeness* of the fundamental and important business operations

which are covered in the model and to decide if new processes have to be added, or process scopes to be extended;

- exploration of *what-if business scenarios* which help to understand and analyse the model, and to help choose from a set of competing appropriate business models (from the case library).

6 Conclusion

Through this work we show how an informally specified business model can be formalised using logic. Having achieved the move from an informal to a formal representation of models and model rules we are able to provide verification and validation support throughout the development lifecycle of a model — making use of techniques such as first order predicate logic and case-based reasoning.

In particular, the introduction of a procedural model extends BSDM with a model execution capability. This adds to the general understanding of how the dynamics of largely informal business models can be captured and used for computer aided model validation.

The use of a tool such as *KBST-BM* provides a number of potential benefits. It provides a means for storing and communicating models. Furthermore, it can increase productivity as much of the model building task is automated, e.g. through the use of a case library. The systematic analysis and checking facilities increase the quality of the model, therefore raising the confidence of the modellers that the model is a good representation of the real world. When a business model is used as a first step in the development of a software system, the improvement of its quality will consequently lead to a better IT system. Similarly, business process reengineering methods can benefit from the improved quality of such models.

References

[1] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide.* Object Technology. Addison-Wesley, February 1999.

[2] Yun-Heh Chen-Burger and David Robertson. *Formal Support for an Informal Business Modelling Method. Proceedings of The 10th International Conference on Software Engineering and Knowledge Engineering, San Francisco, USA*, June 1998.

[3] Yun-Heh Chen-Burger, David Robertson, John Fraser, and Christine Lissoni. *KBST: A Support Tool For Business Modelling in BSDM. Proceedings of Expert Systems 95: Applications and Innovations in Expert Systems III, Cambridge, UK*, December 1995.

[4] Rational Software Corporation. Web site: http://www.rational.com/index.jtmpl, 1999.

[5] Michael Hammer and James Champy. *Reengineering the Corporation: A Manifesto for Business Revolution.* Harper Business, May 1994.

[6] IBM United Kingdom Limited. *Business System Development Method, Introducing BSDM*, 2nd edition, May 1992.

[7] Ivar Jacobson, Maria Ericsson, and Agneta Jacobson. *The Object Advantage - Business Process Reengineering with Object Technology.* Addison Wesley, 1995.

[8] Jintae Lee, Michael Gruninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost, and others. *The PIF Interchange Format and Framework.* The PIF Working Group, 1996. http://www.aiai.ed.ac.uk/project/pif/.

[9] Thomas W. Malone, Kevin Crowston, Jintae Lee, Brian Pentland, Chrysanthos Dellarocas, George Wyner, John Quimby, Charley Osborne, and Abraham Bernstein. *Tools for inventing Organizations: Toward a handbook of organizational processes.* Centre for Coordination Science Massachusetts Institute of Technology, 1993.

[10] Richard Mayer, Christopher Menzel, Michael Painter, Paula Witte, Thomas Blinn, and Benjamin Perakath. *Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report.* Knowledge Based Systems Inc. (KBSI), September 1995. Available at http://www.idef.com/overviews/idef3.htm.

[11] National Institute of Standards and Technology. *Integration Definition for Function Modelling (IDEF0)*, December 1993.

[12] Craig Schlenoff, Amy Knutilla, and Steven Ray. *Proceedings of the Process Specification Language (PSL) Roundtable. NISTIR 6081, National Institute of Standards and Technology, Gaithersburg, MD*, 1997. http://www.nist.gov/psl/.

[13] Mike Uschold, Martin King, Stuart Moralee, and Yannis Zorgios. *Enterprise Ontology. The Knowledge Engineering Review: Special Issue on Putting Ontologies to Use*, 13, 1998. Also available as technical report for AIAI, as AIAI-TR-195.

MODELING STATION DUTY OFFICER OPERATIONS ASSISTANT AT JOHNSON SPACE CENTER

Madjid Tavana, Ph.D.
La Salle University
tavana@lasalle.edu

James N. Ortiz, Ph.D.
National Aeronautics and Space Administration
james.n.ortiz1@jsc.nasa.gov

Susan E. Torney
National Aeronautics and Space Administration
susan.e.torney1@jsc.nasa.gov

Abstract

The Mission Operations Directorate (MOD) at the Johnson Space Center (JSC) is responsible for the planning and conduct of human space flight missions. MOD is challenged to sustain and develop new operations capabilities to support increasingly demanding requirements and to improve its processes to accomplish these missions at higher levels of safety, mission success, and effectiveness. Automation is being considered as an enabling technology to meet the aforementioned challenges. The synergistic combination of flight controllers and intelligent software providing the function of "operations assistants" (OA) is being pursued as the key implementation of this technology in the Mission Control Center (MCC). This paper presents a unique two-stage specification methodology that combines Data Flow Diagrams (DFDs) with Petri Nets (PNs) to design and develop the OA for a complex space station system. DFDs are used to enhance communication with non-technical users while PNs are intended to provide technical designers with a formal basis to rigorously investigate semantic properties of the system.

1. Introduction

Automation technology has been applied to unmanned spacecraft, robots, and ground-based systems at NASA – Johnson Space Center (JSC). However, it remains to be successfully utilized in human mission operations. Operations Assistants (OA), an automation technology at NASA, are a set of distributed customized systems designed to assist the Flight Control Team (FCT) and Flight Director (FD) in the Mission Control Center (MCC). This technology, which is required for exploration programs, is developed for the International Space Station (ISS) program through a series of incremental prototypes. Once these prototypes are matured, they will be transferred to other programs, including the Space Shuttle and Mars exploration. OA maintains flight controller awareness of planned activities and operational objectives (mission cognizance) and monitors, tracks, and checks operational processes, events, and systems. In addition, OA provides just-in-time support tailored to events, alarms, and other problems with intelligent software designed to monitor and retrieve relevant data and analyses. Finally, OA proposes procedures and actions for troubleshooting and recovery where feasible by assessing the conditions, availability, and operability of resources and systems.

This paper addresses the emerging field of modeling and specification used to model a large and complex automation project at NASA. Flight rules and procedures, log events and organization, and system goals and constraints are among some of the knowledge-based data sources used in this study. Large and complex information systems are difficult and expensive to build and maintain. The need for formal system specifications has led to the introduction of several modeling tools. We focus on Structured Analysis to translate user's vague and ambiguous requirements into precise and formal specifications required for system implementation (DeMarco, 1979 and Gene and Sarson, 1980).

Structured analysis assists systems analysts in defining user requirements and developing creative solutions to meet them. Many structured analysis tools have been developed over the past 25 years to aid systems analysts. Data Flow Diagram (DFD), representing information flow, system boundaries, and environmental interactions is a popular structured analysis technique used during the analysis phase of development to communicate with non-technical users (Baylin, 1987). DFD, a popular technique introduced in the late 1970s, is used by most

systems analysts in today's software development projects (Millet, 1999; Kievit and Martin, 1989). The popularity of DFDs stems from its use of intuitively defined concepts and notations. However, DFDs lack a formal basis to rigorously investigate semantic properties of the application. France (1992) has addressed this problem with semantically extended DFD, capable of producing formal specifications. Tao and Kung (1991) address this problem by proposing precedence relation, an abstraction of the functional semantics that specifies the "is-used-to-produce" relationships among the data flows. DFDs also do not represent control logic in systems being modeled. Wards (1986) proposes transformation schema, a notation and formation rule for building a comprehensive system model. Several authors have tried to address the control logic problem inherent in DFDs by integrating DFDs with Petri Nets (PNs) (Richter, G. and Maffeo, B., 1993; Bullers, W., I., 1991; van Hee, K. M., Somers, L.J.A.M., and Voorhoeve, M., 1991). However, these studies do not provide a direct translation of DFDs into PNs for further verification and enhancement of the process model.

Structured analysis emphasizes the description of "what" or the functional aspects of the system. However, a second dimension of modeling complexity, control flow, emphasizing on the describing of "how", is often as important as functional analysis (Yourdon, 1986). Some extensions to structured analysis methodology such as the Structured Analysis and Design Technique (Ross, 1977) and Operational-requirements Approach (Zave, 1982) have considered the explicit representation of control. However, most of these techniques are inadequate for modeling concurrent and asynchronous systems (Murata, 1984).

PNs were initially defined by Carl Adam Petri (Petri, 1966) and later refined and named after him by Holt (Holt, 1971). Peterson (1981) elegantly discusses the dynamic behavior of PNs, while Murata's tutorial review paper provides a thorough review of PN's history and applications (Murata, 1989). PNs and their modifications provide a rich and versatile approach to modeling. They have been proven to be useful for the modeling and analysis of several classes of systems including communication systems (Berthelot and Terrat, 1982), knowledge-based systems (Jantzen, 1980), and process control systems (Bruno and Narchetto, 1986; Camurri et al., 1992). Wang et al. (1991) have used PN's to design a coordination system for intelligent mobile robots. We initially use ordinary PNs here since we are only interested in the control flow aspects of specifications. As the models evolve, coloured PNs could be used to represent attributes associated with tokens.

We present a two-stage specification methodology that combines two independent modeling tools in conjunction: DFD and PN. The proposed methodology, Data Flow Petri Nets (DFPN), is intended to enhance structured analysis by providing process and control specifications that relate the descriptions of "what" and 'how" more closely to the actual system implementation. DFPN is especially useful for systems that may possess concurrent, distributed, asynchronous, parallel, or event-driven qualities. We illustrate DFPN in the context of a SDO OA project. SDOs are particularly good candidates for this methodology since they perform several concurrent tasks during their daily shifts at Mission Control Center (MCC). Next, we illustrate our methodology in the context of a simple supermarket check cashing problem.

2. Data Flow Petri Nets (DFPN) Example

In order to understand the specification techniques utilized here, we begin with an illustrative check cashing example in a supermarket:

> "When the customer has a valid check-cashing card, check will be accepted for the amount of purchase plus $25.00. If the customer does not have a valid check cashing card and the purchase is less than $20.00, two forms of identification must be shown in order to pay by check in the amount of purchase. Otherwise, the store manager must be called to authorize the acceptance of the check for the amount of purchase."

First, we develop the DFD shown in Figure I describing the check cashing problem presented above. DFDs are graphs with four different types of components called *processes*, *data stores*, *external entities*, and *data flows*. A process is presented with a circle, a data store with an open-ended rectangle, an external entity with a square, and data flows are directed arcs connecting processes to processes, data stores, or external entities. External entities are normally shown in the highest level DFD called context diagram. Directed arcs into a process are *input data flows* while directed arcs out of processes are *output data flows*. In order to validate a check cashing card, check cashing and purchase amount data flows are input data flows to card validation process. This validation results in valid or invalid check cashing card. If the customer has a valid check cashing card, check is accepted in the amount of purchase plus $25. Invalid check cashing cards go through purchase amount evaluation. If purchase amount is $20 and more, a manager is called to approve/reject the check. For purchases less than $20, checks are accepted for purchase amount if the customer has two valid Ids. Manager is called to approve checks without two valid IDs.

Insert Figure I Here

Once the DFD is developed, it is converted into a PN. Although, this is currently done manually, we are in the process of building an automated system to facilitate this conversion. In general, all *processes* are changed into *transitions* and all *data flows* are changed into *places*. Mutually exclusive data flows in DFDs cause a process to be broken down into two equivalent transitions in a PNs. PNs are directed bipartite graphs with two different types of nodes called *places* and *transitions*. A place p is presented with a circle and a transition t is presented by a rectangle. The nodes are connected through directed *arcs*. Directed arcs from p to t create *input places* while directed arcs from t to p create *output places*. Each place contains zero or many tokens drawn as black dots. The execution of PN may effect the number of tokens in a place. As it is shown in Figure II, our check cashing PN includes 13 places and 16 transitions. In this example, there might be three tokens; one in place Idle cashier (p_1), one in place check cashing card (p_2), and one in place purchase amount (p_3). The token in p_1 indicates that the cashier is free, the token in p_2 indicates that the customer has a check cashing card, and the token in p_3 shows that the customer has finalized his/her selection and the purchase amount is known. A transition is called *enabled* when each of its input places has enough tokens. A transition can be *fired* only if it is enabled. When a transition is fired, tokens from input places are used to produce tokens in output places. With a token each in p_1, p_2, and p_3, transitions t_1 and t_2 (Validating the check cashing card) are enabled. If the check cashing card is valid, t_1 is fired. Firing t_1 means consuming three tokens, one from p_1, one from p_2, and one from p_3; and producing one token for p_4. Now transition t_4 (Accept check for purchase amount plus $25) is fired producing a token for p_5. Next, transition t_{14} is fired and a token is produced for t_1 which mean the cashier is now free to process the next customer. As long as there are tokens in p_2 and p_3, the two transitions t_1 and t_2 are enabled and this cycle is repeated depending on the specific circumstances. Note that the cashier modeled by this PN processes one customer at a time. Next, we present a detailed description of the SDO DFPN system developed in this study.

Insert Figure II Here

3. Station Duty Officer DFPN

The Station Duty Officer (SDO) performs the lead International Space Station (ISS) operations role during quiescent periods when the FCT and FD are off-duty. This flight controller is responsible for alerting appropriate FCT discipline and the FD if an off-nominal condition develops. The SDO will maintain the radio-frequency command and telemetry link with the ISS via Early Communications Subsystem (ECS) to assess the condition and operability of major station systems such as electrical power, thermal control, life support, communications, attitude control, and data handling systems.

OA, assist SDOs on their tasks dealing with monitoring the status and health of the ISS. The OA will help the SDO maintain an awareness of all the processes performed on board and will assist with the responses to anomalous conditions. The OA for this position will support the concept of reduced control center staffing during quiescent times. SDOs are primarily responsible for two sequential activities, *monitoring* and *handover*. *Monitoring* activities involve concurrent observation and examination of several computer displays in MCC including the Event Logger (ELOG) Display, SDO Display, C&W (Caution and Warning) Logger Display, and EWCA (Emergency, Warning, Caution, and Advisory) Status Display. In addition SDOs listen to various communication loops such as Houston Support Group (HSG) and Ground Control (GC). *Handover* activities are the result of shift changes between FCTs and SDOs or SDOs and SDOs.

A series of DFDs and PNs modules for all monitoring and handover activities are developed in this study similar to the check cashing example described earlier. Once all modules are completed, synthesis in performed to develop an integrated PN for the SDO operations. This is accomplished by identifying overlapping places and transitions and eliminating them. Figure III present the overall PN for SDO operations. The data dictionary for this PN is presented in Figure IV.

Insert Figures III and IV Here

4. Conclusion

This study presents a unique two-stage specification methodology used to develop an OA for the SDO. The SDO DFPN model described in this study is 1) A unique systems specification effort that has not been adopted in previous automation efforts at JSC, 2) It utilizes DFDs and PNs to help users better understand the automation process and validate the data and control flow specifications needed in systems development (elimination of the black box syndrome), 3) It provides users and development teams with a more complete and efficient set of systems specification blueprints, and 4) It helps in sustaining the tool once the system is operational.

References and a complete version of the paper are available upon request from:

Madjid Tavana
Chairman, Management Department
La Salle University
Email: tavana@lasalle.edu

Figure-I: Check Cashing Data Flow Diagram

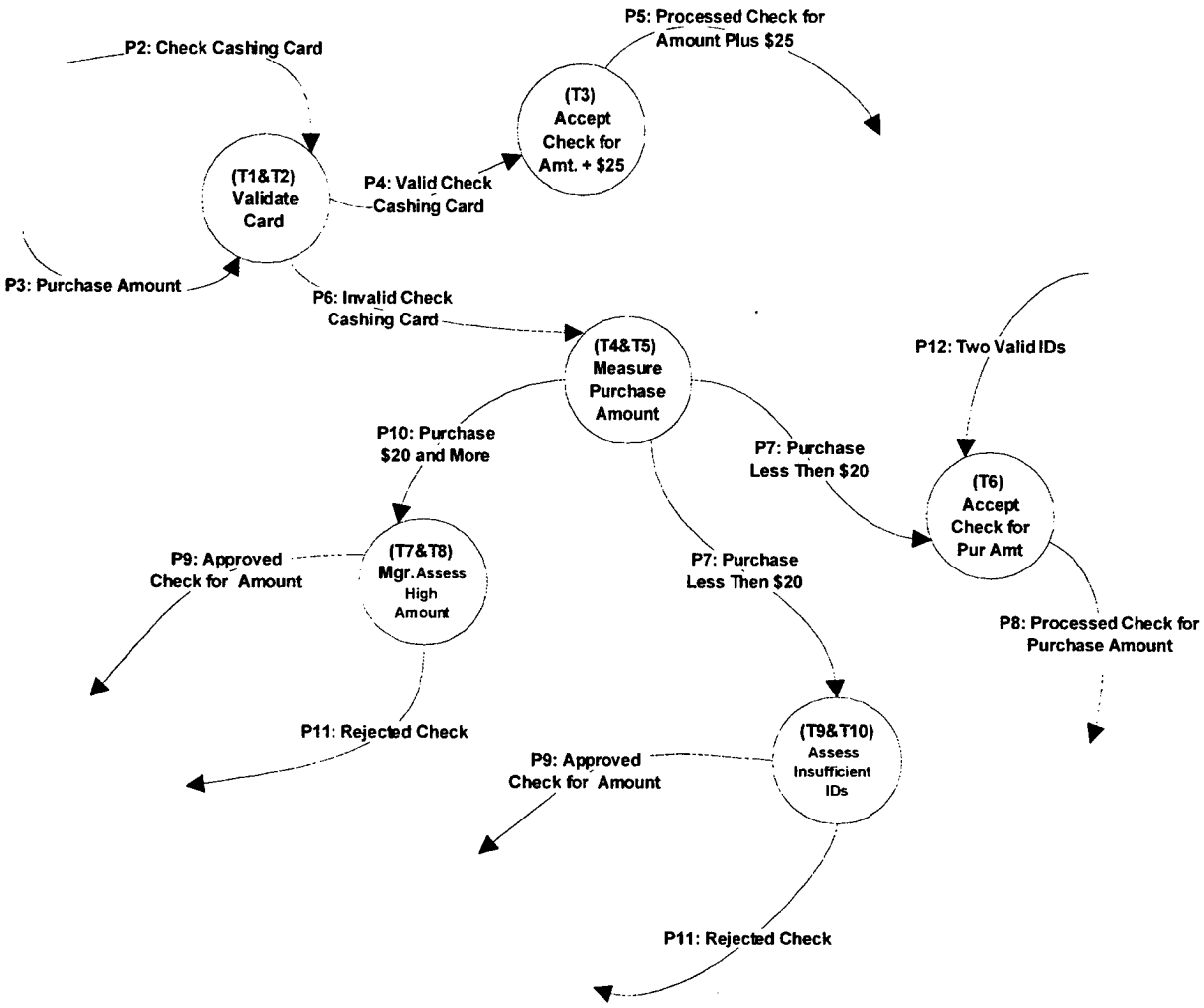


Figure-II: Check Cashing Petri Net

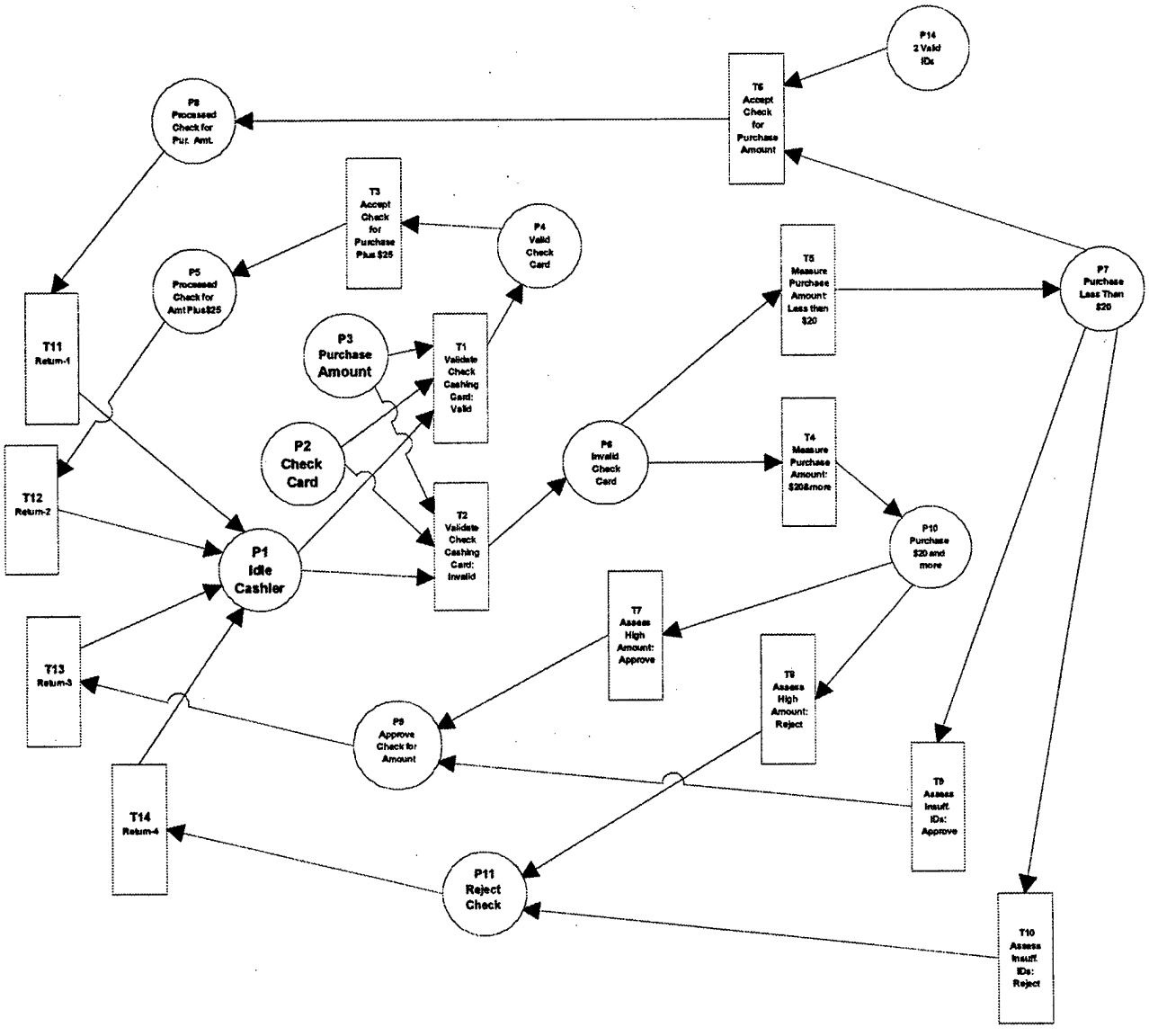


Figure-III: SDO Overall Petri Net

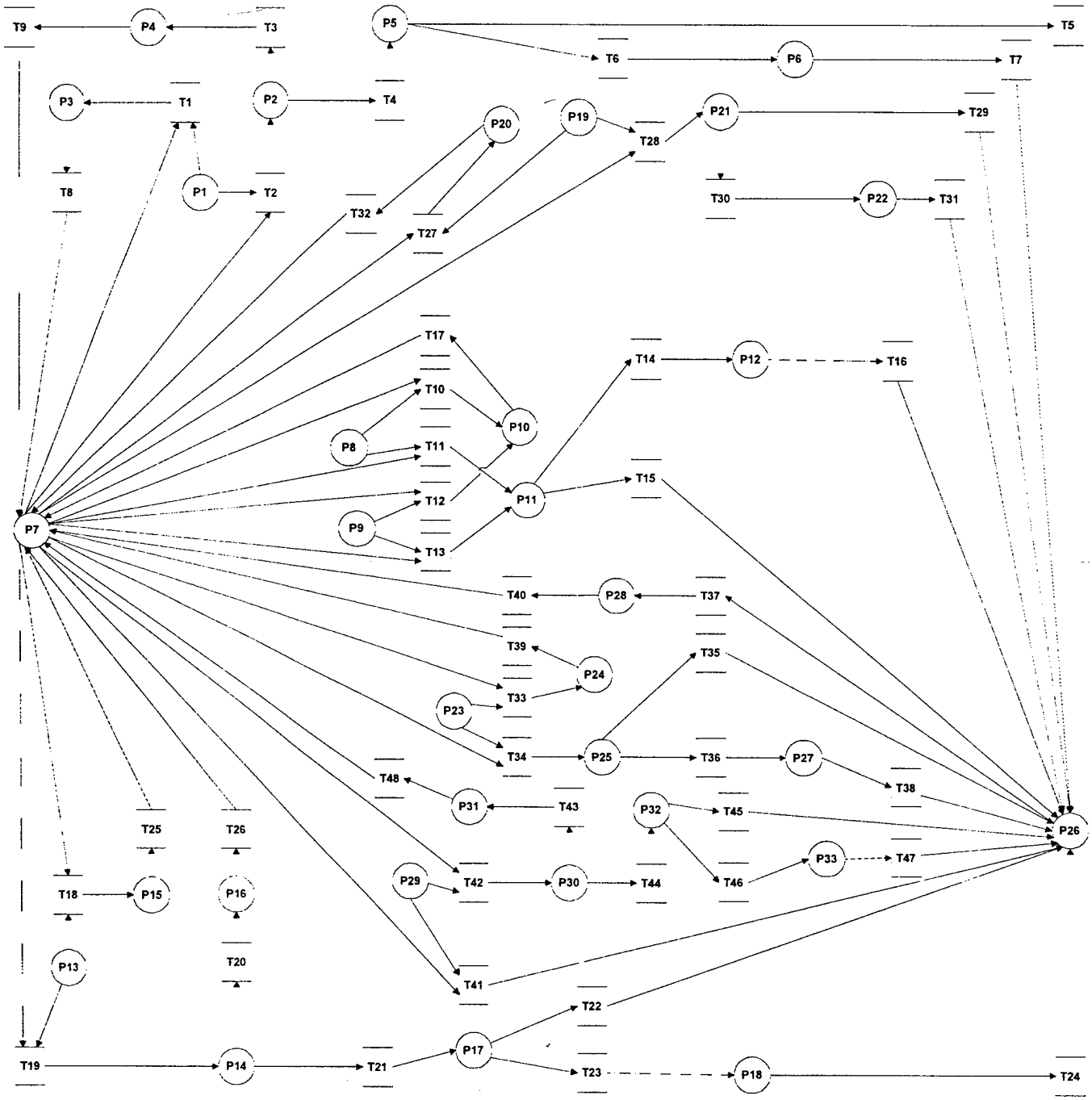


Figure-IV: SDO Overall Petri Net Data Dictionary

Transitions		Places	
T1	Initial Assessment:Regular ELOG	P1	ELOG Message
T2	Initial Assessment:Irregular ELOG	P2	Irregular ELOG
T3	Display Verification:Non-Anomalous ELOG	P3	Regular ELOG
T4	Display Verification:Anomalous ELOG	P4	Non-Anomalous ELOG
T5	ELOG Anomaly Assessment:Non-Critical	P5	Anomalous ELOG
T6	ELOG Anomaly Assessment:Critical	P6	Critical ELOG
T7	Call FCT/FD-ELOG	P7	IDLE SDO
T8	Return	P8	EWCA C.I.
T9	Return	P9	C&W Message
T10	C&W Display Ver:Discarded C.I.	P10	Discarded Warning
T11	C&W Display Ver:Anomalous C.I.	P11	Anomalous Warning
T12	EWCA Display Ver:Discarded C&W	P12	Critical Warning
T13	EWCA Display Ver:Anomalous C&W	P13	COMM Telemetry Data
T14	Warning Anomaly Assessment:Critical	P14	Deviated COMM
T15	Warning Anomaly Assessment:Non-Critical	P15	Expected COMM
T16	Call FCT/FD-C&W	P16	Non-Anomalous COMM
T17	Return	P17	Anomalous COMM
T18	COMM Comparison:Expected	P18	Critical COMM
T19	COMM Comparison:Deviated	1P9	HSG COMM
T20	COMM Investigation:Non-Anomalous	P20	Routine HSG COMM
T21	COMM Investigation:Anomalous	P21	Anomalous HSG COMM
T22	COMM Anomaly Assessment:Non-Critical	P22	Critical HSG COMM
T23	COMM Anomaly Assessment:Critical	P23	GC COMM
T24	Call FCT/FD-COMM	P24	Routine GC COMM
T25	Return	P25	Anomalous GC COMM
T26	Return	P26	Logging Message
T27	HSG COMM Assessment:Routine	P27	Critical GC COMM
T28	HSG COMM Assessment:Anomalous	P28	Responsive Measure
T29	HSG Anomaly Assessment:Non-Critical	P29	OAL Activity
T30	HSG Anomaly Assessment:Critical	P30	Deviated OAL
T31	Call FCT/FD-HSG	P31	Non-Anomalous OAL
T32	Return	P32	Anomalous OAL
T33	GC COMM Assessment:Routine	P33	Critical OAL
T34	GC COMM Assessment:Anomalous		
T35	GC Anomaly Assessment:Non-Critical		
T36	GC Anomaly Assessment:Critical		
T37	Log Message		
T38	Call FCT/FD-GC		
T39	Return		
T40	Return		
T41	OAL Activity Comparison:Expected		
T42	OAL Activity Comparison:Deviated		
T43	OAL Activity Investigation:Non-Anomalous		
T44	OAL Activity Investigation:Anomalous		
T45	OAL Anomally Assessment:Non-Critical		
T48	OAL Anomaly Assessment:Critical		
T47	Call FCT/FD-OAL		
T48	Return		

Modeling Agile Command and Control Innovations for Joint Air Operations

Rick G. Goodwin
Science Applications International Corporation
Richard.G.Goodwin@cpmx.saic.com
Dr. Paul E. Girard
Science Applications International Corporation
Paul.E.Girard@cpmx.saic.com

Abstract

Revolutionary improvements in access to information will enable rapid, agile and precise command and control (C2) of military forces. Joint Air Operations is a large-scale enterprise control system where information technology advancements are effecting large-scale improvements in offensive and defensive joint air operations against sophisticated symmetric and asymmetric threats. Modeling and control of non-linear, time varying processes within a large enterprise of autonomous and semi-autonomous entities requires intuitive methods for representing the physical and abstract aspects of the enterprise as well as technologies where complex dynamic phenomena can be described using analytical methods or empirical rules. DARPA is exploring a number of different control approaches such as game theory, model predictive control, Kalman filtering, neural nets, and fuzzy petri nets as new techniques and models that can be applied as smart procedures and functions within the Joint Air Operations enterprise. This paper presents SAIC's approach to developing an Air Operations Enterprise Model that facilitates simulations and experiments of multiple advanced C2 controls within a complex enterprise model and includes dynamic phenomena such as unexpected failures and enemy attacks on C2 assets.

1. Introduction

SAIC is developing an Air Operations Enterprise Model (AOEM) that incorporates advanced theoretical techniques and tools for agile control of military operations. The purpose of SAIC's AOEM is to support experimentation and analysis of controller and 'plant' behaviors by capturing the appropriate level of metrics to evaluate complexity of physical entities, stimulus and timeliness of decision elements, and agility and stability of the system. A secondary purpose of the AOEM is to support the JFACC System Architect in designing and assessing alternative C2 architectures to leverage the advances in agile control for military operations. This paper will describe our early efforts at defining the AOEM components, how we plan to incorporate new control techniques and algorithms within the AOEM, and a description of our modeling applications and knowledge bases that will support the experimentation and analysis of new models.

SAIC's AOEM modeling environment includes technology to implement decision elements with embedded controls for reasoning under uncertainty, estimate states with incomplete information, introduce non-linear dynamic phenomena, predict future states and system reliability, and gateway products for accessing new control techniques and algorithms that are external to the AOEM framework.

2. Enterprise Model Components

As illustrated in Figure 1, the AOEM provides a representation of the environment, physical assets, sensors to observe assets, effectors to control them, communications to convey information, and Command and Control Nodes (C2Nodes) to process information and perform Command and Control functions. New control theories and algorithms will be embedded within C2Nodes supporting Joint Air Operations that include both continuous and discrete elements that will be modeled as a dynamic hybrid system.

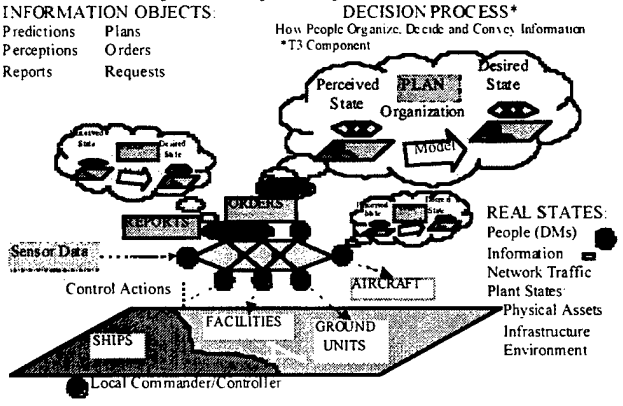


Figure 1. Joint Air Operations Enterprise Model

SAIC's AOEM approach is a knowledge-based and expert system approach towards modeling the AOEM components and embedding new C2 controls that rely primarily on discrete, symbolic, and continuous variables. The AOEM hybrid control model will allow the exploration and analysis of alternative C2 architectures that include controllers with the agility to intelligently adapt and react to stochastic events, dynamic changes to plant structure and processes, and dynamic phenomena which disrupts and destroys. SAIC's AOEM modeling environment is a comprehensive, object-oriented (OO) environment for building and deploying mission-critical, intelligent applications that dramatically improve complex enterprise operations.

The AOEM includes the necessary C2 components of an air operations plant and Decision Elements (DE) with the required model fidelity to stimulate and evaluate the new control techniques and modules. The AOEM processes include flow of information between C2Nodes and components in the AOEM enabling the simulation and assessment of new controllers with regards to how controls handle such dynamics as incomplete, inconsistent, inaccurate, and late information. The AOEM will also enable simulation of disturbances to processes and information flows to analyze whether the system can converge to a stable state and the predicted impact on downstream processes and plans.

AOEM plant dynamics (in this case, air operations and tactical plans executed by JFACC units) are initiated by one or more JFACC objectives and manifest themselves in scenarios. Simulations, as representations of plant dynamics, are periodically interactive or asynchronously interactive in order to accommodate the control inputs they receive and feedback information they send [1].

2.1 Environment and Physical Asset Components

Within the AOEM, environment and assets will have attributes of physical state including terrain and weather. Physical Assets represent Entities at several levels of description: "Units" consisting of smaller Units (providing an unlimited tree-hierarchy for physical organization structure), plus additional Platforms and Components (attached to platforms or amassed with a Unit) expressed in the aggregate, or, when necessary, as individual instances. All Units are treated equally in the model, not in separate "sides" (red and blue). Each Unit has an affiliation and will all be capable of sensing and attacking others, even their own "friendlies". Capabilities of Assets have "true" values in the plant model, but are treated as "perceived" values in the decision model. The difference between true values and perceived values will be discussed further in section 2.4.

As is illustrated in Figure 1, the Physical Assets included in the Enterprise Model include such things as units, platforms, communications, weapons, sensors, orders, reports, plans, sensor data, C2Nodes, and controls.

2.2 Sensors, Effectors, and Communications

Sensors, Effectors and Communications have both physical and information states. Sensors will provide information in the form of observations of physical states. Effectors will convert control information into physical state and actions. Communications will be represented as Assets, and as capabilities to exchange messages among Decision Elements, such as, likelihood of delivery, and delay, if delivered, and, if necessary, with processing and transmission rates.

As an example of sensor data that will be modeled at the tactical level, the sensor data will include such attributes as time-of-arrival of a platform at a specified destination. The model will also include capability and effectiveness data such as a platform's ability to reach a certain destination within a specified time window with the appropriate capability (payload, range, etc.) and appropriate probability of success.

Some experiments will be performed to examine the effect of sensors on state estimates and decisions. The AOEM will include sensors in such a way that "efficiency" of the sensor can be tuned with a dial that will range from a perfect "all seeing" sensor to one that exhibits errors, uncertainty, and delay in various functions.

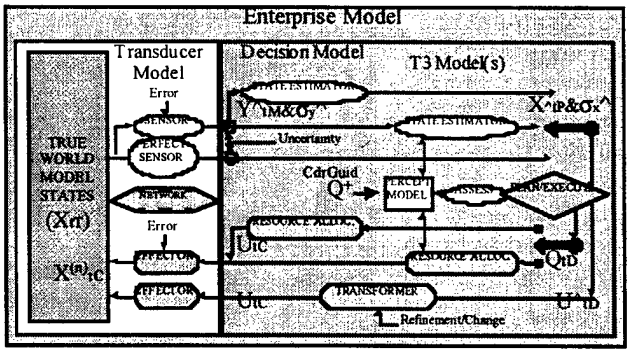


Figure 2. Truth, Perception, Decision, Control Model

For the discussion which follows regarding Figure 2, symbol X = true state, Y = measurements, X^ = estimated state, U = controls, and Q = directives. As illustrated in Figure 2, for baseline operations, a perfect, all seeing, all-knowing sensor capability will be achieved, which can be queried at any time. Otherwise, rates of reporting and filtered outputs, such as, "only enemy aircraft", may be specified. These provide the measurement states, Y. The new control modules may include state estimation algorithms, which will need the measurements, Y. For those that do not, the Enterprise

Model will provide state estimation, X^, with or without errors.

To allow scenario dynamics to be controlled, each Unit will be controllable though a C2Node. This will provide movement control and control of activity, such as engagement. The variety and structure of controls, U, will be implemented as required to support experiments of new controllers. To govern decision processes, some new control modules will need directives, Q, from a superior, such as objectives, tasks, activities, priorities, and rules of engagement. In a strategy-to-task hierarchy, time-based priorities are encapsulated and provide an inherited element to support weight-of-effort and priorities down to the lowest level.

2.3 Command and Control Nodes

Command and Control (C2) Nodes "know" information states. To support experiments with the effects of distributed decision-making the AOEM will include Communications processes to pass information objects between C2Nodes that are not collocated. These processes will indicate if and when a message is delivered and if that message is corrupted or degraded in the process. For baseline operations, instant and perfect delivery will be modeled and provided. As shown in Figure 3, "Communications Among Decision Elements", for each T3 control module, the Enterprise Model will include C2Node processes needed to support that module, or other modules belonging to that C2Node.

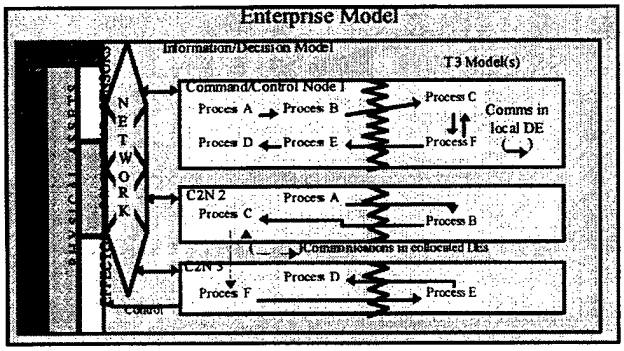


Figure 3. Communications among Decision Elements

The jagged line defines the interface between the C2Nodes and the controllers. The interface will pass data that is logically within the C2Node. Communications assets within the AOEM, incurring potential delays, disruptions and corruption along the way will send information needing to reach another C2Node. The controllers will direct the C2Nodes within the AOEM to send and receive information and perform actions within the enterprise as appropriate.

Using the illustration in Figure 3 as an example, a control function desires to change the route of an airplane.

a) If the control function would reside in the affected aircraft (C2N 3 in Figure 3)- the controller would interface directly with the aircraft object. There would be no modeling of a communications network required.
b) If the control function would reside in a superior C2Node (e.g. an Airborne Command Post) (C2N 2 in Figure 3) - the controller would interface directly with C2N 2 which would then communicate with the affected aircraft via the Enterprise Model communications network. C2N 2 -> C2N 3
c) If the control function would reside within the Command Center (C2N 1 in Figure 3)- the controller would interface with C2N 1. The communications network would be utilized in a manner specified by the current organizational structure. E.g. C2N 1 -> C2N 2-> C2N 3 or C2N 1 -> C2N 3.

If C2N 2 and C2N 3 are collocated, the communications indicated by dashed lines may represent a LAN with instant or delayed communications. [1]

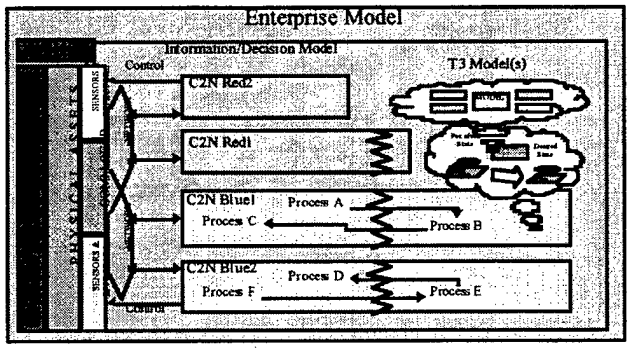


Figure 4. Enemy and Own Decision Elements

As illustrated in Figure 4, controllers and decision elements will exist within the AOEM to control enemy components just as they exist to control "own" forces. Figure 4 illustrates the fact that information objects will be passed between the enemy C2Nodes and their controllers in the same way they are for own forces.

2.4 Information Exchange Between Control Modules and AOEM Components

Working with BBN, SAIC is helping define the order-of-battle and types of units, platforms and components including a description of the features, capabilities and effectiveness of particular types as a reference for all instances, notional or discrete. (Notional instances are represented as an aggregate number of a type (or template) of that class, while discrete instances are individually created in the model and have separate identity.) Attributes and capabilities of an asset, such as (perceived)

maximum speed are found on the (perceived) type (or template) of that class of asset, while current speed (estimate) is on the (reported) instance or unit containing the instance (notional or discrete). When capabilities need to be expressed relative to another asset, an effectiveness table will provide these data. The C2Nodes may each have their own perception, i.e., their own copy, of these type definitions, capabilities, and effectiveness tables, which can be passed over the interface to the new control modules.

3. SAIC's Object-Oriented Expert System Enterprise Model Development Environment

SAIC is using Gensym Corporation's G2 product as a highly interactive and visual model development environment that simplifies and speeds the prototyping, development, and deployment of the AOEM. G2 objects provide an intuitive way to represent the physical and abstract aspects of the AOEM. C2 objects are organized in a hierarchical class structure, and G2 provides the flexibility of multiple inheritance so that a C2 element can inherit properties and behaviors from multiple C2 object classes. Once an object – or class of objects – is defined, the models can be immediately reused. Any object or group of objects can be cloned repeatedly, and each cloned copy will inherit all the properties and behaviors of the original object. Objects, rules, and procedures can be grouped into library modules that are shared by all G2 models, allowing streamlined development of new models.

This approach minimizes AOEM development efforts by capturing knowledge about C2 components within the model using generic rules, procedures, formulas, and relationships that apply across entire classes of objects. Expert knowledge about AOEM objects is being expressed using rules. One property that is being defined in C2Nodes is that of connectivity. Connection descriptions can be defined for AOEM objects as inputs outputs or bidirectionals. Each object connection can be referred to as an individual object and can include heuristics and rules that govern the behavior of the connection. Rules may be invoked by forward chaining (event and data detection) and backward chaining (goal and data seeking). Natural language syntax is used for capturing heuristics in rule objects. In addition, more complex heuristics within the AOEM are captured in procedures or methods – which allow a more sophisticated algorithms and techniques for intelligent reasoning.

Another form of (non-visual) connectivity we are using within the Joint Air Operations Enterprise Model is that of relations. Connections modeled between objects enable reasoning based on relationships such as "subordinate-to", "attached-to", "part-of", "loaded-in", etc. Such relationships carry a logical, but non-visual connectivity between objects.

3.1 Joint Air Operations Enterprise Model Development & Experimentation Environment

Using G2 Telewindows, JFACC T3 (tool, technique and theory) developers and Enterprise Modelers can collaborate over the same model, experiment with their control models collaboratively, and evaluate their technology with a distributed team of Subject Matter Experts, the System Architect, and the Enterprise Modeling team. The models can be experimented with remotely and an Enterprise Model that integrates multiple control technologies can be displayed simulataneously by multiple, remote users with Telewindows. The JFACC team can evaluate actual-to-ideal performance and perform "what-if" analysis with assets and controllers in the AOEM to help optimize and tune emerging designs and architectures.

3.2 Simulating Dynamic Battle Space Conditions

Our enterprise modeling team is simulating dynamic battle space conditions by incorporating stochastic models of the environment and enemy behavior within the AOEM model. C2Nodes will include decision functions used to reason about real-time behavior and "focus" on the appropriate knowledge for the decision at hand. Communication of sensor data as well as directives can experience delays or degradation, such as delays in reporting critical enemy locations, battle damage assessment, or critical target lists to subordinate commanders. Units and platforms can represent complex dynamic behavior such as physical motion and aggressive behaviors for responding to stochastic events. C2Node functions will exist in the model to reason about such things as rates of change, standard deviation of values over time, and other time-based functions for evaluation and assessment of the controllers.

3.3 Layered Technologies and Toolkits on G2

SAIC's AOEM will be developed using layered technologies and toolkits for implementing the AOEM components, connections and relationships among components, rules that govern these connections, methods and procedures to support complex reasoning about the state, and implement advanced decision elements within the AOEM while evaluating advanced hybrid control models. The layered applications and knowledge bases include neural networks and fuzzy logic, statistical tools, genetic algorithms, reactive and supervisory control models, and bayesian networks. The following sections provide a brief discussion of these layered technologies and how they may be applied to the AOEM.

3.3.1 Modeling Dynamic, Non-linear Phenomena. Modeling controllers and component behaviors that provide agility in the face of dynamic, non-linear phenomena requires models that ensure data quality, include important process variables, diagnose state, implement control functions and optimally effect resources. Neural networks can be used to recognize and forecast disturbances, detect and diagnose faults, combine data from redundant sensors, perform statistical quality control, and adaptively tune process controllers. Back-propagation networks can be used to implement function mapping networks that implement a functional relationship by presenting the network with inputs and target outputs and, by applying a learning algorithm, teach the network the relationships. Radial basis function networks are effective for solving multiple-class membership problems such as distinguishing between infrastructure failures and enemy attack disturbance events. Rho networks are ideal for single-class membership problems detecting whether a specific data pattern describes a state that is a member of a certain class. Auto-associative networks can operate as a type of signal processor.

SAIC will use the G2 NeurOn-Line™ toolkit for embedding neural networks within the AOEM as appropriate to support intelligent monitoring, state estimation, and decision element control tasks. Information monitoring tasks are modeled using signal filtering, sensor validation and soft sensors with auto-associative and back propagation nets. State estimation tasks will use fault detection and diagnosis models using Rho and radial basis function nets. Advanced decision element controls can be modeled using inverse-model adaptive control with back-propagation nets.

3.3.2 Enterprise Model Connectivity Solutions. SAIC's AOEM connectivity solutions include object-oriented bridge products and tool kits that allow G2 applications to interface with other systems including leading databases, Programmable Logic Controllers, Distributed Control Systems, data managers, and the Internet. Existing bridge products within SAIC's AOEM development environment include ODBC, CORBALink, ActiveXLink, and JavaLink Gateway products to integrate and embed access to controllers and scenario and model data sources into the AOEM.

3.3.3 Alternative Business Process Designs. SAIC's AOEM is being developed in such a way to make it easy to visualize complex air operation processes. The SAIC model development environment allows automatic simulations to benchmark the performance of new control techniques and technologies within the enterprise processes. "What if" analysis is used to test re-engineering ideas with questions like, "How will embedding a new control module within the Joint Air Operations enterprise impact operational cost and cycle-time?" and "What utilization can be achieved with additional or current resources?" After re-engineered processes have been implemented with advanced control techniques, the JFACC team can use the design model on-line to form the basis for valuable Joint Air Operations intelligent control applications.

SAIC's AOEM modeling team is using Gensym's ReThink™ for the graphical design, simulation and analysis of new controllers within Joint Air Operation processes. With ReThink™, our modelers are working together with Subject Matter Experts and T3 developers to explore alternative process designs using hands-on modeling and simulation software tools. Graphical blocks are connected to describe the sequences and interdependencies among processing tasks. These blocks are being arranged in a number of different combinations to describe AOEM processes at multiple levels of detail.

3.3.4 Uncertainty Reasoning. Controllers and C2Nodes within the AOEM will require models for sensor validation and data fusion as well as diagnosis and prediction of problems providing agility in the system at the edge of chaos from environmental stochasticity that arises from more-or-less unpredictable interactions with the outside world. SAIC AOEM development environment includes Gensym's BayesOn-Line™ (BOL) software tool for reasoning about uncertainty and for learning relationships among variables using a graphical modeling technology popularly known as Bayesian networks. Using domain knowledge and statistical data about C2 processes, models can be constructed and used to predict values of some variables given the observation of others. The BayesOn-Line™ tool set is useful for instantiating controllers and decision elements that perform sensor validation, diagnosis, prediction, decision analysis, risk analysis, success analysis, monitoring and correlation.

3.3.5 Generic Fault Propagation and On-Line Diagnosis. Experimentation plans will be developed by our EM team to capture controller and plant reactions to effectively evaluate the new controllers in the midst of a dynamic, battle space environment. The values and metrics to be collected during the experiments require monitors on many different events in simulation-time. There is a need to filter out redundant events, correlate events, diagnose the root cause of problems, execute tests when there is evidence of a problem, and execute automated and manual corrective actions to address the symptoms, and capture these results. These captured metrics and values will automatically be communicated to other C2 components and be used to develop probabilistic models of system behaviors. Gensym's NeurOn-Line® and GDA can operate together to provide model-based reasoning for diagnostics of air operations.

Our AOEM modeling team and subject matter experts define the "lines of reasoning" used in the diagnostic reasoning and associated explanations using the GDA

graphical object-oriented interface. Causal directed graphs will be developed that provide a methodology for simulation-time fault assessment, addressing the derived requirement of problem identification based on symptoms, diagnostic testing, and fault isolation.

4. Conclusion

For SAIC's enterprise-modeling team, G2 and these layered products will be reducing our time-to-completion of the AOEM, and is minimizing project risks. G2 enables our modelers to represent knowledge about joint air operations as objects, rules, methods, and procedures using graphics and structured natural language allowing models to be readily understood, tested, and easily modified by the JFACC team.

SAIC's AOEM development environment based on G2 provides the JFACC team with a powerful, distributed; open toolkit for developing and conducting integrated experiments. These experiments will allow powerful collaboration by distributed participants when evaluating the agility and stability of new control techniques and modules within the air operations enterprise.

5. References

[1] BBN and SAIC, "Interface Object Model Supporting Data Exchange Between the Enterprise Models and T3 Modules", *DRAFT Version 1.0 Program Document*, November 1999, pp. 1-5.

[2] Logica Carnegie Group, "System Architect Requirements for the AOEM", *DRAFT Version 1.0 Program Document*, October 25 1999, pp. 1-4.